Unpack Local Model Interpretation for GBDT

Wenjing Fang, Jun Zhou, and Xiaolong Li

Ant Financial Services Group, Hangzhou, China
{bean.fwj,jun.zhoujun,xl.li}@antfin.com

Abstract. Gradient boosting decision tree(GBDT), which aggregates a collection of single weak learners (i.e. decision trees), is widely used for data mining tasks. Because GBDT inherits the good performance from its ensemble essence, much attention has been drawn on the optimization of this model. As with its popularization, an increasing need for model interpretation arises. Besides the commonly used feature importance as a global interpretation, feature contribution is a local measure that reveals the relationship between a specific instance and the related output. This work focuses on the local interpretation and proposed a unified computation mechanism to get the instance-level feature contributions for GBDT in any version. Practicability of the mechanism is double checked by the listed experiments and utilizations in our real scenarios.

1 Introduction

Machine learning have great success in modeling data and making predictions automatically. In many real-world applications, we need more than the black-box model and an explanation is needed. For example, when customers apply for the loan on credit, the loan officers will compute their credit scores based on their historical behaviors. In this case, it's far from enough to only show customers the final score and the loan officers should convince them with detailed reasons. The most efforts made in data mining is mainly on improving the accuracy and efficiency, which result in excellent models. However, little attention is drawn to model interpretation for these models. Several common measures for the variable significance have been proposed. Gini importance is a commonly used importance measure for Random Forest, which is derived from the Gini index[2]. Gini is used to measure impurity between the parent node and two descendent nodes of samples after splitting. The final importance is cumulated from the Gini change for each feature over all the trees in forest. This general feature importance(FI), also known as *global interpretation* shows the important factors of the target, which unpack the general information in trained models. But it doesn't take any feature values of an instance into consideration which is not enough in some ways. *Local interpretation*, on the other hand, places particular emphasis on a specific case and reveals main cause of each record. This type of interpretation makes up for the shortages of global one. A method is raised to define the feature contributions(FC) [13] ,which is cumulated from label distribution changes, as a description of the feature impact on the output. The value of feature contribution

reveals how much a feature contributes and the sign represent whether it's a positive influence or not.

GBDT[7] is an ensemble model built on top of a bunch of regression decision trees. It has some appealing characteristics. For example, GBDT can naturally handle nonlinearity and tolerate missing values. As a winning model in many data mining challenges [8, 1, 3], GBDT is a good option for regression, classification and ranking problems with well-known generalization ability. Besides its wide range of applications, GBDT is also flexible in allowing users to define their own suitable loss functions. Furthermore, there are many implementations[4][9] and much work has been done to speed up the training process. For GBDT, global feature importances calculation is widely used to do the feature selection. For example, Breiman proposed a method to estimate feature importance[7]. First, it make an approximation of relative influences of a split as the empirical improvements in squared-error. By summing over all the non-terminal nodes in a decision tree for every split feature, a cumulative importance is obtained. For a collection of boosting trees, the averaging feature importance among them is the global relative feature importances. In this way, the changes of loss function are divided into splitting features and obtain a feature measure related to the training process.

Given the popularity and high quality of GBDT, it's important to uncover internals of the model. Existing work has largely ignored the exploration of local interpretations, which will be the focus of this paper. Specifically, we will study feature contributions for GBDT. We starts from former procedures of model interpretation for random forest[13] and update the definition of the feature contribution. The proposed mechanism is flexible enough to interpret all versions of GBDT. The original definition based on label distribution change is prove to be a special case of ours under a particular loss function.

The rest of the paper is organized as follows. Section 2 provides a brief review of related work on local interpretations. Section 3 gives out the formal definition of feature contribution as preliminary and presents the approach for calculating feature contributions for random forests. In section 4, we describes the rationale as well as main actions to interpret GBDT. Section 5 contains experiment settings and the process to examine the proposed methodology. At the end, section 6 concludes the work presented.

2 Related Work

Model interpretation provides convincing reasons to the model outputs. In the occasion when a detailed explanation is needed for a prediction, even a reduction in model performance is acceptable. This is the reason why some analysts turn to logistic regression for help. Logistic regression is a typical linear model with thorough study. It is likely that researchers will examine the regression coefficients , which is global feature importance after the normalization preprocess. The coefficient weights of different features are in the same scale, so that the relative value between coefficients represents the relative importance between features.

Because of its linearity, the instance-level feature contribution is easy to compute as the product of actual feature value and its coefficient. Both the global and local interpretation of logistic regression is transparent from observing the formula and reflect the impact on the function value of independent variables. Linear models is regarded as interpretable with good quality and speed. In the field like credit scoring, it is the default choice.

Another type of solutions prefer the good performance of complex models and interpretability of simple models. The pipeline of this type will first make use of advanced models as a black-box and then extract useful information out of it with the help of a more interpretable model. For example, [5] is an interesting attempt to conduct the model interpretation for neural network. The proposed learning-based method tries to retrain a decision tree that approximates a trained network in order to get inductive rules. And a novel approach in [6] formally treat the interpretation of additive tree models as extracting the optimal actionable plan. It models the optimization problem as an integer linear programming and utilize existing toolkit as the solver. The constraints are based on both tree output score and the objective function. Finally, the linear functions over variables for leaf node flags and feature value spans are get and they could be efficiently solved. Notice that, these two kinds of methods need extra training process especially for the interpretation and bring new models or tasks to solve.

Some other researchers come up with model-independent local interpretations. They mainly make changes to feature value and test the chain effect to performance loss of predictions. The loss is then taken as the measure of local importance of feature. This method only rely on the output evaluation and provide a unified way to check feature contribution for black-box models. Leave-One-Covariate-Out(LOCO) [12] share the same spirit of feature importance computation of random forest [2]. By replacing the actual feature values with missing, zero or average values, the impact of a feature in predicting is then removed. The instance-level contributions of all the features can be calculated separately and compared with each other. Moreover, this method is also work for global feature importance.

Different from the models with a continuous closed-form functions, tree-based models are discrete. Directly calculating the gradient as the feature change is not suitable for this kind of models. As a derivative of decision tree, the random forest goes further on model interpretation than GBDT. The method in [11] introduces the way to compute the feature contributions so as to show informative results about the structure of model and provide valuable information for designing new compounds. The work in [13] extends this idea to support explanation for categorical predictions. These interpretations customize the definition of feature contributions as the label distribution changes and cumulate these contributions according to the path of an instance. This method makes full use of the information, not only the training data but also the model structure. It is natural to design the interpretations with the model structures to get a more reasonable result.

In most cases, GBDT outperforms linear models and random forest. This work proposes an easy way to get the feature contribution on the instance-level. The interpretation process make full use of model structure and extra model is not need. Generally, it can be apply to all versions of GBDT implementations with little preprocessing and modification to the prediction process.

3 Preliminary

Additive tree models are a powerful branch of machine learning but are often used as black boxs. Though they enjoy high accuracies, it's hard to explain their predictions from a feature based point of view. Different ensemble strategies bring out different models while sharing the tree structure as a basis. So the model interpretations for different addictive tree models share some key spirits and can spread out from one to another with appropriate adaptation. In this section, we first review a practical interpretation method for random forest and introduce the general definition of feature contribution to better illustrate the proposed model interpretation for GBDT.

3.1 Interpretation for Random Forest

Random forest is one of the most popular machine learning models on account of its superior accuracy utilizing categorical or numerical features on regression and classification problems. A random forest is a bunch of decision trees that are generated respectively and vote together to get a final prediction. Every tree is trained on randomly sampled data and subsampling feature columns to introduce the diversity for better generalization, which is the key weakness of single decision tree models. Random forest is known as a typical bagging model and the bagging strategy works out by averaging the noises to get a lower variance model.

The process to generate a random forest from a given dataset is shown in algorithm 1 and 2. The training process generates a forest with M trees based on dataset D and function $splitData$ divide current dataset into two parts according to the split feature and split value. While predicting a new instance X_i, each tree in $Forest$ first votes for one class and a final prediction is then concluded by the majority. Function $goLeft$ tells whether the instance falls into left branch of current decision subtree. Algorithm 1 is the utility for decision tree for both random forest and GBDT. Trees grow gradually as described and there is a pair of splitting feature and splitting value at every branch of a single tree. They are chosen according to pre-defined $Gain$ which measures the improvement of a split. $getLeafWeight$ will return either a class or score and the computation is determined by the model.

An instance starts a path from the root node all the way down to a leaf node according to its real feature value. All the instances in the training data will fall into several nodes and different nodes have quite different label distributions of the instances. For simplicity, we only show the binary classification here and

it can be extended to multi-classification. Every step after passing a node, the probability of being the positive class changes with the label distributions. All the features along the path contribute to the final prediction of a single tree.

A practical way to evaluate feature contributions is explored[13]. The key idea is taking the distribution change values for the positive class as the feature contribution. Concretely, it takes four procedures to work:

1. Computing the percentage of positive class of every node in a tree
2. Recording the percentage difference between every parent node and its child nodes
3. Cumulating the contributions for every feature on each tree
4. Averaging the feature contribution among the trees in the forest

The method consists of an offline preparation embedded in train(steps 1-2) and an online computing with the prediction process(step 3-4). It is easy to record the local contribution(or local increment) and related split feature to every edge in a tree. In the algorithm 2, the positive class percentage in $D_{k,s}$ could be computed while entering function $BuildTree$. With an extra parameter $parent$, we can compute the percentage difference between this node and its parent. Next, record this local contribution in the node information and pass this node as a parent node when $BuildTree$ is called to build subtrees recursively. Finally, every node except the root of a tree retains a local contribution of the split feature in the parent node and the algorithm will store this additional information in model file. As for the prediction, we only have to read the pre-computed local feature contribution of the nodes that a new instance passes through and aggregate them as the definition, which won't take much extra time.

3.2 Gradient Boosting Decision Tree

GBDT is another type of ensemble model contains a collection of regression decision trees. However, the ensemble is based on gradient boosting which promotes the prediction gradually by reducing the residual. For every iteration, a new model is built up to fit the negative gradient of the loss function until it converges under an acceptable threshold. And the final prediction is the summation of all stagewise models. Gradient boosting is a general framework and different models are available to be embedded. GBDT introduce decision tree as the basic weak learner. When square error is chosen as the loss function, the residual between current prediction and target label is the negative gradient which is computational friendly.

From the above definition, we can see the differences between random forest and GBDT, some of which are the main obstacles to stop us from employing the model interpretation of random forest to GBDT:

1. Random forest aggregates trees by voting, while GBDT sum up the scores from all the trees. This means that the trees in GBDT are unequal and the trees have to be trained in sequential order. The interpretation should make proper adaptations to deal with this problem.

Algorithm 1 Decision Tree

```
function BuildTree(D_{k,s})
    if all samples in D_{k,s} are in the same class or have the same features then
        node = new Node()
        node.isLeaf = True
        node.score = getLeafWeight(D_{k,s})
        return node
    end if
    for each feature q ∈ S do
        for every split value p ∈ split(q) do
            D_left, D_right = splitData(D_{k,s}, q, p)
            compute the gain G_{q,p} = Gain(D_{k,s}, D_left, D_right)
        end for
    end for
    choose the split(p,q) = argmax_{q,p} G_{q,p}
    node = new Node()
    node.isLeaf=False
    node.split=(q, p)
    node.left = BuildTree(D_left)
    node.right = BuildTree(D_right)
    return node
end function
function TreePredict(X_i,root)
    if True == root.isLeaf then
        return root.score
    else
        if True == goLeft(X_i,root.split) then
            return TreePredict(X_i,root.left)
        else
            return TreePredict(X_i,root.right)
        end if
    end if
end function
```

2. Decision tree in GBDT outputs a score instead of a majority class type for classification problems. Though we can get the label distribution changes as random forest interpretation, this score should be wisely taken into consideration.

3.3 Problem Statement

Given a training dataset $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$, where N is the total number of training samples, $x = (x_1, x_2, ..., x_S)$ implies a S dimensional feature vector, $x^{(i)}$ is the feature vector for the i-th sample and $y^{(i)}$ is the related label. We can illustrate training process of GBDT as in algorithm 3. r_{mi} is the residual for sample i in the m-th iteration.

Algorithm 2 Random Forest

```
function TRAIN(D,M)
    Init Forest = {}
    for m = 1, 2, ..., M do
        Bootstrap samples: randomly select k samples from D as D_k
        select s variables at random of D_k as D_{k,s}
        T_m =BUILDTREE(D_{k,s})
        Forest = Forest ∪ T_m
    end for
    return Forest
end function
function PREDICTINSTANCE(X_i,Forest)
    Init class set C = {}
    for each T_m ∈ Forest do
        C_m = TREEPREDICT(X_i, T_m)
        C = C ∪ C_m
    end for
    choose the class r with most predictions
    return r
end function
```

Besides the basics of model, we introduce the notation of feature contribution by denoting the model interpretation of random forest in section 3.1 :

$$LI_f^c = \begin{cases} Y_{mean}^c - Y_{mean}^p, & if\ the\ split\ in\ the\ parent\ is\ performed \\ & over\ the\ feature\ f \\ 0, & otherwise \end{cases} \tag{1}$$

LI_f^n in equation 1 is the Local Increment(LI) of feature f for node n defined before. For binary classification, Y_{mean}^n represents the percentage of the instances belonging to the positive class in node n.

$$FC_{i,m}^f = \sum_{c \in path(i)} LI_f^c \tag{2}$$

$$FC_i^f = \frac{1}{M} \sum_{m=1}^{M} FC_{i,m}^f \tag{3}$$

On a single tree m, $FC_{i,m}^f$ in equation 2 cumulates the feature contribution of feature f for a specific instance i. Equation 3 later average all the feature contribution for feature f among all the trees.

4 Mechanism

Looking back at model interpretation for random forest, its central spirit is to establish the idea of feature contribution. By computing label distribution,

Algorithm 3 Gradient Boosting Decision Tree

function TRAIN(D,M)
  Init $f_0(x) = 0$
  for $m = 1, 2, ..., M$ **do**
    Compute residual:
    $r_{mi} = y_i - f_{m-1}(x_i),\ i = 1, 2, \ldots, N$
    Train a regression decision tree from residual:
    T_m =BUILDTREE(D)
    Cumulated prediction sum:
    $f_m(x) = f_{m-1}(x) + T_m$
  end for
  Get finally boosting function:
  $f_M = \sum_{m=1}^{M} T_m$
  return f_M
end function
function PREDICTINSTANCE(X_i,f_M)
  score = $\sum_{m=1}^{M}$ TREEPREDICT(X_i, T_m)
  return score
end function

a measure of the change is then got and associated with the split feature. In the case of GBDT, we can expand this computation with a slight modification. Because the targets of latter trees are the residual, it should be the replacement while computing label distribution. Nevertheless, the problem of this version is that the label distributions on leaf nodes are not always equal to the score on it. So the valuable information of these scores are not utilized and the method cannot reflect different GBDT versions [7][4].

In fact, the loss function determines the optimal coefficient and table 1 shows some common examples. LS and LAD stand for Least Square and Least Absolute Deviation respectively. $\tilde{y}_i$ is the residual updated after each iteration. $F_{m-1}(x_i)$ is the approximation on iteration $(m-1)$. g_i and h_i are the first and second order gradient statistics on the loss. Different from the numerical optimization essence to compute negative gradient(for LS and LAD), XGB first approximate the loss function with its second order Taylor expansion and an analytic solution is then got. So it contains no negative gradient computation and the evaluation of leaf weights is far from the label distribution. Particularly, only if the LS loss function and traditional GBDT training process is used, the label distributions meet the scores.

Without loss of generality, the interpretation for GBDT need to work on the leaf scores. Since the scores are only assigned to leaf nodes, we have to find a way to propagate them back all the way to the root. The left tree of Fig 1 shows an example tree in a GBDT model, with split feature and split value marked on arcs. Observing the three nodes in the rounded rectangle, the instances in node 6 will get a score difference as: $S_{n11} - S_{n12} = 0.085 - 0.069 = 0.016$, where S_{n11}

Table 1. Loss Functions of GBDT

Settings	Loss Function	Negative Gradient	Leaf weight
LS	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$	$ave_{x_i \in R_{jm}} \tilde{y}_i$
LAD	$\mid y_i - f(x_i) \mid$	$sign[y_i - f(x_i)]$	$median_{x_i \in R_{jm}}\{y_i - F_{m-1(x_i)}\}$
XGB	$\sum_{i=1}^{n}[l((y_i, \hat{y}^{(t-1)})) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i))] + \Omega(f_t)$	/	$-\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$

is the score on node 11. Moreover, this difference is caused by splitting feature $feat5$ branching by a threshold of 1.5. We can allocate this difference to the two branches by assigning the average score of child nodes to their parent node. For instance, $S_{n6} = \frac{1}{2}(S_{n11} + S_{n12}) = \frac{1}{2} \times (0.085 + 0.069) = 0.0771$. Then, the local increment metrics could be calculate with the scores, $LI^{n11}_{feat5} = S_{n11} - S_{n6} = 0.085 - 0.0771 = 0.0079$. Similarly, the leaf scores as well as the local increment could be spread to the whole tree.

The interpretation process while predicting is the same as that of the random forest. On the right hand side of Fig 1, all the node average scores and feature contributions on the tree is marked. Supposing an instance gets a final prediction on leaf node 14 of tree t, a cumulation through the path: $n0 \rightarrow n2 \rightarrow n5 \rightarrow n9 \rightarrow n14$ will be executed: $FC^t_{feat5} = LI^{n2}_{feat5} = -0.0201$, $FC^t_{feat2} = LI^{n5}_{feat2} = -0.0073$,$FC^t_{feat4} = LI^{n9}_{feat4} + LI^{n14}_{feat4} = -0.0015 + 0.0010 = 0.0025$.

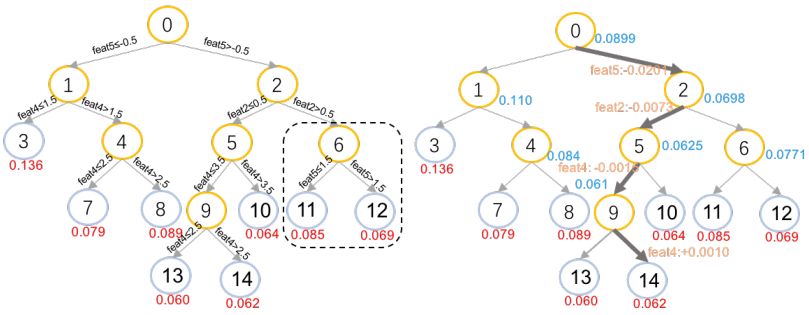

Fig. 1. Feature Contribution Example for GBDT

Thinking over the propagation strategy, the average score is assigned to the node 6 which assumes an instance falls into the left branch or the right with equal probability. So the expectation of intermediate nodes could be revised as in equation 4:

$$S_p = \frac{1}{2}(S_{c1} + S_{c2}) \rightarrow \frac{N_{c1} \times S_{c1} + N_{c2} \times S_{c2}}{N_{c1} + N_{c2}} \tag{4}$$

Where the N_{c1} and N_{c2} is the number of the instances fall into child nodes node c1 and c2. These statistics need extra information from training process.

By viewing the computation in this brand new way, we get a flexible interpretation mechanism by only use the leaf node scores and instance distributions, regardless of the implement settings of GBDT. Under the setting of the LS loss function, we can notice that not only the label distribution meets the prediction score on leaf node but also label distribution of the intermediate nodes meets our back propagated scores. That is to say, the label distribution method is a special case of our mechanism with this particular settings. And this method also support the multiple classification problems.

5 Experiment

In this section, we demonstrate the experiments on the proposed interpretation. In the first place, we show the mechanism is reliable with respect to global feature importance. Then we compare our interpretations to that of random forest and find it accord with the global feature importance better. Finally, we study the interpretation at real cases in our scenario and get a satisfied analysis for them.

5.1 Experiment settings

The GBDT version in our experiment is the Scalable Multiple Additive Regression Tree (SMART)[14], which is a distributed algorithm under parameter server. Hundreds of billions of samples with thousands of features could be trained by the algorithm. Not only the storage usage but also the running time cost is optimized without the loss of the accuracy. The training data is drawn from transaction table under the scene of Fast Pay(FP) in Alipay. The transaction is marked as a positive if it is reported as a fraud by the customer. To keep a balanced ratio between positive and negative cases, normal transactions only retain 1% by random sampling.

Predictive Model Markup Language(PMML)[1] is an XML-based predictive model interchange format. PMML restricts a standard expression of different models and allows analysts share model among different prediction implementations. PMML include the process of data pre-processing and post-processing along with model prediction. The structure of PMML files follows the order to predict an instance, the compulsory parts of tree models are:

1. Header: general information about PMML document.
2. Data Dictionary: field definitions in model.
3. Model: contains the definition of the data mining model.
4. Mining Schema: a list of all fields used in the model.
5. Output: name all the desired output fields expected from the model.

[1] http://dmg.org/pmml/v4-3/GeneralStructure.html

Fig. 2. GBDT Model in PMML Format

Fig 2 is a toy fraction of GBDT PMML file and the tree embedded in it can be translate as the tree in Fig 1. The element *Node* is an encapsulation for a tree node, which contains a predicative rule to choose itself or its siblings. The attribute *id* assigns a unique number to each node in a tree. The value of *score* in a *Node* is the predicted value for an instance falling into it. *SimplePredicate* is a simple boolean expression indicating the split information. Our pre-trained model is stored as a PMML file. JPMML[2] is employed as the evaluator and we change it slightly to print the nodes on the prediction path for an instance.

5.2 Consistency check

We implement the feature contribution as previous description in [7]. In order to make the interpretation to be independent on the training process of GBDT, we experiment without change the training algorithm. Because the distribution of instances is needed according to equation 4. JPMML will predict the training instances and record instance distributions on every nodes. According to the tree structure in model and instance distributions, a pre-process is done by back propagating the local increments as shown in section 4. With the local increments, the feature contributions of the new instances is then computed . After interpreting lots of instances, we can get a distribution of feature contributions

[2] https://github.com/jpmml/jpmml-evaluator

among the instances. The median is a robust estimator for the expectation of the general feature contribution and should somehow keep accordance with the global feature importances metrics.

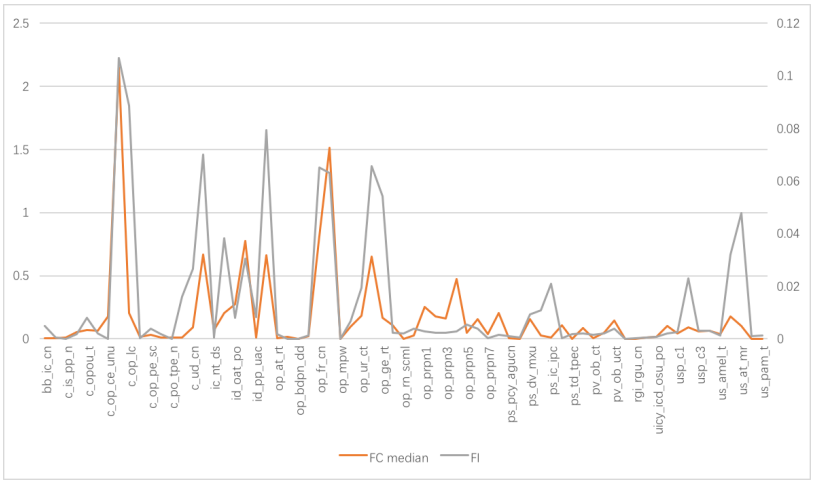

Fig. 3. Feature Importance and Feature Contribution Medians

Fig 3 plots the Feature Importance(FI) and Feature Contribution(FC) medians for every feature. As we can see, this two statistics have similar relative value and are in good agreement. It proves that the interpretation for GBDT is practical and reasonable.

5.3 Comparison to Random Forest

Following the experiment of last section, we get a ranking of the feature contribution median. This ranking is a measure of feature importance and reflect the quality of local interpretation. We implement the work for random forest in [13] and compare it with our ranking. For justice, we replace the GBDT feature importance with Information value (IV) as the importance metric. IV is a concept from information theory and show the predictive strength for the target label[10].

In Fig 4, we compute the intersection size on different variable coverage(i.e. Top 10-50 features of IV). *RF* implies the method explained in section 3.1. *GBDT* is the simple average strategy with only the information in PMML file. *GBDTV2* is the modified version in equation 4. From the result, our interpretations better captured the importance and the revised version works best.

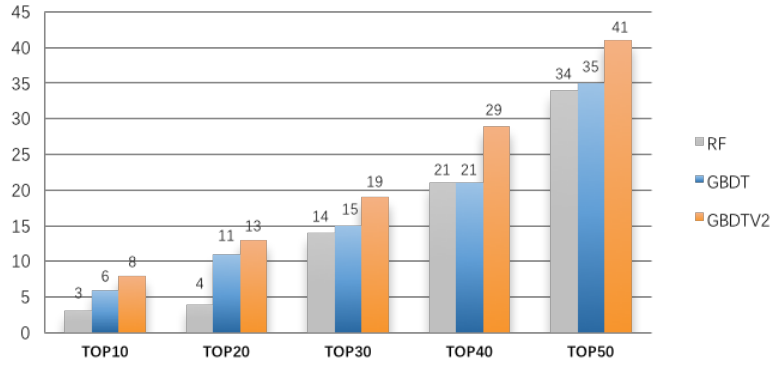

Fig. 4. Interpretation: GBDT v.s RF

5.4 Case Study

Besides the general evaluation, we analysis the 300 specific instances. Fig 5 shows an example case, we only list some representative fields and divide them into 4 parts. The variables are ranked by IV(general feature importance) and checked the feature risk manually by domain experts. We can draw the following conclusions:

- Part I: Variables in this section are with high IV, our interpretation is able to capture the features that are judged to be high risk(marked as blue fields). The feature with high IV but low risk (judging from the feature value) is assigned a lower score, so the interpretation is good for instance-level contributions.
- Part II: There are 2 variables(colored pink) with high IV and marked high risk is missed by the interpretation, which mainly due to its low occurrence in split features. The global importance of these two variables is also low and model interpretations are limit by the model.
- Part III: Variables with median or low IVs are not caught by mistake and is assigned a low feature contribution for that case.
- Part IV: Several variables are considered to be high risk for the particular instance, even the general IVs of them are low. Our interpretation find them out and show the superiority of the local feature contributions over the global feature importance.

Further more, if we conduct interpretations on a batch of fraud cases which are missed by the model, the local feature contributions will mine valuable rules and help analysts improve the model.

Variables	IV	IVrank	Feature Importance	Feature Value	Risk	
c_op_ud_lk	2.203	1	0.106791601	2	High	I
opcd_pn_cy	1.762	2	0.065620865	0	High	
id_pp_cy	1.525	3	0.030499262	0	High	
op_st_at_d	1.445	4	0.079317491	1	High	
op_fd_pob	1.315	5	0.063102101	-1	Low	
ic_dcn_dr	1.126	6	0.07008353	0.333333333	High	
op_fr_cn	1.065	7	0.065047636	1	High	
c_op_lc	0.92	8	0.088589539	0	High	
op_ur_ct	0.874	9	0.019405404	2	High	II
id_oat_po	0.842	10	0.008055243	1	High	
c_op_cr_y	0.811	11	0.002003042	2	High	
uicy_icdcy_ud	0.778	12	0.00064423	0.078567	Low	III
c_opcd_cn	0.693	13	0.00193592	0	Median	
uicy_icd_osu_po	0.68	14	0.000859095	0.117214	Low	
ps_pn_ct	0.651	15	0.00016962	15	Low	
op_prpn1	0.624	16	0.002749283	0	Low	
ic_ay_dcn	0.618	17	0.038145332	-1	Low	
us_bste_t	0.608	18	0.03190312	0	High	IV
usp_c1	0.343	29	0.002563547	4	High	
op_ge_rt	0.203	52	0.054343817	0	High	
us_at_mr	0.196	53	0.047708771	56.50257164	High	

Fig. 5. Case Study for Interpretation

6 Conclusion

Employing models as a black-box is not enough. A measure for the impact of a feature on the prediction convinces analysts in an intuitive way. The local interpretation provides an explanation when necessary and contributes to the develop of the models. We describe the method to unpack the interpretation for the advanced model GBDT. To the delight of analysts, the whole process is independent from the training details and technical optimizations. Only the tree structure and instance distribution are needed, which can be easily extracted by a post-processing after training. The label distribution based method of random forest is proved to be a special case of our method. We explore the distribution of local feature contribution and prove it to be in agreement with global feature importance. The method is applied to real case studies in different scenarios and serves as a good translator of our models.

References

1. Bennett, J., Lanning, S., et al.: The netflix prize. In: Proceedings of KDD cup and workshop. vol. 2007, p. 35. New York, NY, USA (2007)
2. Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)
3. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Proceedings of the Learning to Rank Challenge. pp. 1–24 (2011)
4. 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. pp. 785–794. ACM (2016)

5. Craven, M.W., Shavlik, J.W.: Using neural networks for data mining. Future generation computer systems 13(2-3), 211–229 (1997)
6. Cui, Z., Chen, W., He, Y., Chen, Y.: Optimal action extraction for random forests and boosted trees. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 179–188. ACM (2015)
7. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
8. He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., et al.: Practical lessons from predicting clicks on ads at facebook. In: Proceedings of the Eighth International Workshop on Data Mining for Online Advertising. pp. 1–9. ACM (2014)
9. 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: Advances in Neural Information Processing Systems. pp. 3149–3157 (2017)
10. Kullback, S.: Information theory and statistics. Courier Corporation (1997)
11. Kuz'min, V.E., Polishchuk, P.G., Artemenko, A.G., Andronati, S.A.: Interpretation of qsar models based on random forest methods. Molecular informatics 30(6-7), 593–603 (2011)
12. Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R.J., Wasserman, L.: Distribution-free predictive inference for regression. Journal of the American Statistical Association (just-accepted) (2017)
13. Palczewska, A., Palczewski, J., Robinson, R.M., Neagu, D.: Interpreting random forest models using a feature contribution method. In: Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on. pp. 112–119. IEEE (2013)
14. Zhou, J., Cui, Q., Li, X., Zhao, P., Qu, S., Huang, J.: Psmart: Parameter server based multiple additive regression trees system. In: Proceedings of the 26th International Conference on World Wide Web Companion. pp. 879–880. International World Wide Web Conferences Steering Committee (2017)