RIFF: Inducing Rules for Fraud Detection from Decision Trees

Lucas Martins[1], João Bravo[1], Ana Sofia Gomes[1], Carlos Soares[2], and Pedro Bizarro[1]

[1] Feedzai, Portugal
[2] Faculdade de Engenharia da Universidade do Porto, Portugal

Abstract. Financial fraud is the cause of multi-billion dollar losses annually. Traditionally, fraud detection systems rely on rules due to their transparency and interpretability, key features in domains where decisions need to be explained. However, rule systems require significant input from domain experts to create and tune, an issue that rule induction algorithms attempt to mitigate by inferring rules directly from data. We explore the application of these algorithms to fraud detection, where rule systems are constrained to have a low false positive rate (FPR) or alert rate, by proposing RIFF, a rule induction algorithm that distills a low FPR rule set directly from decision trees. Our experiments show that the induced rules are often able to maintain or improve performance of the original models for low FPR tasks, while substantially reducing their complexity and outperforming rules hand-tuned by experts.

Keywords: Fraud Detection; Rule Induction; Decision Trees

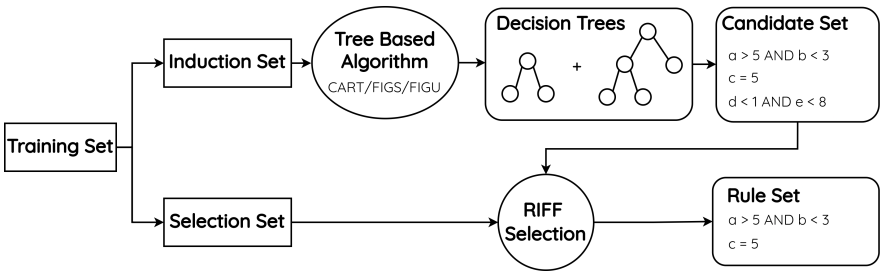

Fig. 1. RIFF Overview

1 Introduction

Despite the advent of modern machine learning (ML) algorithms, rule systems continue to be important in many domains [1, 20]. Their simplicity and interpretability, often requirements in high stake problems, as well as their longstanding presence has earned the trust of many financial institutions. Many continue

to use rule systems as their only solution for fraud detection, while others use them alongside machine learning models.

However, building rule sets traditionally requires expert input and their predictive performance is typically worse than modern machine learning models. This could potentially be attributed, at least in part, to the fact that rules are not automatically inferred from data, but instead manually created and tuned.

While there are several induction algorithms that infer rules from data [17, 4, 19, 20, 16, 13, 10, 5, 6], applying them to fraud detection can be problematic due to the extreme class imbalance that is often present, and the requirement to have very low FPR values, typically under 2%. The latter is necessary as incorrectly flagging legitimate transactions can cause friction, eroding customer trust, leading to financial losses, and putting undue pressure on manual reviewers. Another challenge is to induce rules that are easily understood by experts, which is a requirement in fraud detection for two reasons. Firstly, experts often times need to review the decision made by a rule, and, as such, they must understand its reasoning. Secondly, experts need to manually modify rules periodically to keep up with new fraud patterns.

Our main contribution is a rule induction algorithm, RIFF, that leverages decision trees to build low FPR rule sets for fraud detection. As illustrated in Figure 1, RIFF builds rules by inferring them from data, in order to save expert time spent on maintaining rule sets and analyzing fraud. We benchmark RIFF against state-of-the-art decision trees algorithms, CART and FIGS, and against expert made rules. Our experiments use both publicly available and private real world transaction data, and we benchmark RIFF using trees generated by CART, FIGS, and by our own modified version of FIGS, FIGU. Our results show that induced rule sets created by RIFF outperform a rule system hand-tuned by experts. Plus, RIFF's rules are shown to maintain or improve performance of the original decision tree models, while substantially reducing their complexity.

2 Related Work

Prior work on rule set induction can be divided into two distinct approaches. *Separate-and-conquer algorithms*, also known as covering algorithms, form rule sets by adding rules one by one until a stopping criterion is met [14, 15, 5, 18, 7, 9]. They typically rely on a heuristic to choose the best rule to add, removing all examples covered by that rule going forward.

On the other hand, *divide-and-conquer algorithms* such as ID3 [17], C4.5 [19] and CART [4], use decision trees to describe the data. These trees are grown in a greedy fashion, by iteratively splitting a current leaf node based on the value of one attribute to maximize a chosen criterion, such as *information gain*. FIGS [20] expands on these algorithms (namely CART) by introducing the option of adding a new tree by splitting on a new root node instead of an existing leaf node. Each tree independently contributes to the model with a score that is summed to produce the final prediction.

3 Rule Induction for Fraud Detection

Rule systems used in the context of fraud detection are typically composed of tens to hundreds of simple rules, each designed to capture a particular fraud pattern. These rules are usually a conjunction of a small set of logical conditions that can be understood by a human and evolved with time by tuning specific thresholds. Fraud detection systems are also usually constrained to operate with an overall low False Positive Rate (FPR) or Alert Rate (AR). This is to limit the friction caused to legitimate users or to limit the total number of alerts generated according to the capacity of a fraud analyst team.

Decision tree algorithms like CART have formed the basis of state-of-the-art algorithms for tabular data when used in ensembles such as Random Forests [3] or Gradient-Boosted Decision Trees [8]. However, even single decision trees can be hard to understand and interpret by humans and they can't be manually tuned by experts. We thus propose leveraging these algorithms to generate candidate rules with good discriminative performance. For this, our proposed algorithm, RIFF, is split into two steps (see Figure 1):

1. We induce a set of candidate rules from the leaves of a tree-based model trained on an *induction set*. This candidate set would, ideally, contain different rules with high precision, corresponding to leaves with high purity of fraud examples.
2. The best performing rules out of this candidate set are selected in a greedy fashion based on their performance in an out of sample *selection set* to yield a small set of rules that work well together.

Over the next two sections we describe in detail these two steps.

3.1 Rule Induction from Decision Trees

In order to generate a low FPR rule set, we extract rules from decision trees. We base this decision on the fact the typical splitting criterion tries to find the *purest* leaves, i.e., leaves with highest precision that in theory maximize the amount of gained recall per FPR. For this reason, we will assume that all rules in extracted candidate set predict the *positive* class.

After creating a tree with suitable, high purity leaves, we form a candidate rule set by extracting one rule for each leaf. We do this by traversing the path from the root node to each leaf and by forming a new rule with the conditions in this path. Figure 2 shows an example where a tree model with 5 leaves was converted to a rule set.

This method does not extend well to additive tree models like FIGS, because it ignores tree scores when converting leaves to decision rules. To choose the best split on a tree $i \in [T]$, FIGS uses a mean squared error criterion with residuals calculated by subtracting from the label the predictions from all other trees j as targets. For a sample $(\mathbf{x}, y)$, the residual, r_i, for tree i is thus given by:

$$r_i(\mathbf{x}, y) = y - \sum_{j \in [T], j \neq i} \hat{y}_j(\mathbf{x}) \ , \tag{1}$$

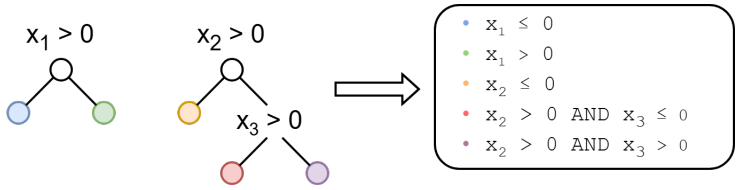

Fig. 2. Extracting rules from a FIGS model

where $\hat{y}_j$ is the prediction for tree j. This additive approach means that leaves generated by FIGS may not be pure enough to yield low FPR rules since they are meant to complement the predictions made by other trees.

For this reason, we modify FIGS by binarizing the residual computation, thus turning its Greedy Tree Sums into Greedy Tree Unions. Concretely, when considering how to split a current leaf node in a tree i, we discard any samples that fall into the support of that node if they are already covered by a current leaf node of another tree j. I.e., we discard a sample $(\mathbf{x}, y)$ when evaluating the splitting criterion if:

$$\bigvee_{j \in [T], j \neq i} \hat{y}_j(\mathbf{x}) = \texttt{true} \ , \tag{2}$$

where $\hat{y}_j(\mathbf{x})$ is now a binary prediction for tree j that evaluates to `true` if: 1) $\mathbf{x}$ falls into the support of a current leaf in tree j with high enough precision (as specified by a user provided threshold); 2) it falls into the support of the current best leaf of tree j as measured by precision. We call this modified version FIGU in the sequence.

3.2 Rule Selection

As mentioned in Section 3, the rule selection step aims to distill a potentially large set of candidate fraud rules into a smaller set, maximizing the number of fraud cases captured, i.e., the True Positive Rate (TPR) of the system, while keeping its FPR or Alert Rate below a given threshold. For concreteness, we will focus on the former constraint, FPR, in the exposition.

Writing as $\text{cov}(R; \mathcal{D})$ the example set covered by a rule set R on dataset $\mathcal{D}$, we have:

$$\text{TPR}(R) = \frac{|\text{cov}(R; \mathcal{D}^+)|}{|\mathcal{D}^+|} \ , \qquad \text{FPR}(R) = \frac{|\text{cov}(R; \mathcal{D}^-)|}{|\mathcal{D}^-|} \ ,$$

where we denote by $\mathcal{D}^+$ and $\mathcal{D}^-$ the subsets of positive and negative examples respectively. We can thus formalize the rule selection goal as choosing a subset of rules S from a given set of candidate rules $C = \{c_1, c_2, \ldots, c_n\}$ to solve:

$$\max_{S \in 2^C} \quad \text{TPR}(S) \quad \text{s.t.} \quad \text{FPR}(S) \leq \text{FPR}_{\max} \ . \tag{3}$$

Both TPR and FPR are monotone non-decreasing submodular functions and this optimization problem is NP-hard [11]. We therefore propose a simple greedy

heuristic algorithm that iteratively selects rules with the highest precision in the remaining uncovered samples, until a stopping condition is met. In our case, we stop when the FPR of the selected rule set surpasses $\text{FPR}_{\max}$. We assume that this always occurs over the runtime of the algorithm, i.e., that $\text{FPR}(C) \geq \text{FPR}_{\max}$.

Algorithm 1 Greedy Rule Selection Algorithm

Input: $(C = c_1, c_2, \ldots, c_n);\ \text{FPR}_{\max};\ \mathcal{D}$
$\mathcal{D}' \leftarrow \mathcal{D}$
$S \leftarrow \{\}$
$i \leftarrow 0$
while $\text{FPR}(S; \mathcal{D}) < \text{FPR}_{\max}$ **do**
  $i \leftarrow i + 1$
  $r_i \leftarrow \arg\max_{r \in C \setminus S} \text{Precision}(r; \mathcal{D}')$
  $\mathcal{D}' \leftarrow \mathcal{D}' \setminus \text{cov}(r_i; \mathcal{D}')$
  $S \leftarrow S \cup \{r_i\}$
end while
return $r_1, \ldots, r_i$

This algorithm returns a list of rules in the order they were selected. Defining $S_i \coloneqq \{r_1, \ldots, r_i\}$ we have that S_{l-1} is guaranteed to satisfy the $\text{FPR}_{\max}$ constraint, whereas S_l may violate it, with l denoting the length of the returned list.

To compare rule sets with different FPR values generated from separate candidate sets, we relax our solution set to include randomized rule sets. Instead of outputting a fixed subset S of our candidate set, we output a probability for every rule in C to be selected. All rules, except the last rule, are thus selected with probability 1. However, for the last rule selected, r_l, this probability is chosen to match the expected FPR with the desired FPR constraint. Formally, the probability $\rho(c)$ for a rule $c \in C$ to be selected is:

$$\rho(c) = \begin{cases} 1 & c \in S_{l-1} \\ \frac{\text{FPR}_{\max} - \text{FPR}(S_{l-1})}{\text{FPR}(S_l) - \text{FPR}(S_{l-1})} & c = r_l \\ 0 & c \notin S_l \end{cases},$$

In practical terms, this means that if a sample is only covered by the last rule, r_l, there is a probability $\rho(r_l)$ of it triggering, therefore influencing the system's decision. In other words, the rule is only checked for a random subset of examples. With this in mind, we can interpret the TPR of this randomized rule system as a random variable with an expected value given by:

$$\text{TPR}(\rho) = (1 - \rho(r_l))\text{TPR}(S_{l-1}) + \rho(r_l)\text{TPR}(S_l)\,.$$

4 Experiments

We evaluate RIFF on two public classification datasets: BAF [12], a synthetic bank account fraud dataset, and Taiwan credit [2], a credit card default dataset.

We also use a private dataset, containing real transaction fraud data, which we cannot disclose due to privacy and contractual reasons. A baseline unique to this dataset is a set of rules manually tuned by data scientists allowing us to compare the rules generated by RIFF against rules handcrafted by experts. An overview of the used datasets can be seen in Table 1.

Table 1. Dataset Analysis Summary. The train/validation/test splits are time-based for the BAF and Industry datasets and random for Taiwan Credit.

	BAF	**Industry**	**Taiwan Credit**
Task	Account Fraud	Transaction Fraud	Credit Card Default
Positive rate	1%	7%	22%
#samples	1M	3.5M	30K
#features	32	113	25
Train split	75%	60%	60%
Validation and test split	12.5%	20%	20%

We split the training set into two smaller subsets: induction and selection. After using the induction set to train CART, FIGS and FIGU models, we extract candidate rules from the generated tree models, as described in Section 3.1. We then apply the selection step of the algorithm to extract the best rules from each candidate set according to their performance on the selection set. We use the validation set to tune the total number of splits used when training the decision-tree model, using a line search over the values [10, 20, 30, 40, 50] and the test set for the final evaluation of the generated rule set.

We use LightGBM as a strong baseline for predictive performance as a state-of-the-art Machine Learning algorithm for tabular data. We also report the performance of the best CART and FIGS models trained in the induction step as divide-and-conquer baselines.

Table 2. Recall at 1% FPR in the test split for BAF, Credit and Industry Datasets

	BAF	**Industry**	**Taiwan Credit**
LightGBM	0.252	0.531	0.084
Expert Rules	-	0.158	-
CART	0.160 $\pm$0.005	0.315 $\pm$0.075	0.063 $\pm$0.009
CART + RIFF	**0.184** $\pm$0.006	**0.362** $\pm$0.027	**0.139** $\pm$0.018
FIGS	**0.210** $\pm$0.006	**0.394** $\pm$0.032	0.067 $\pm$0.016
FIGS + RIFF	0.158 $\pm$0.016	0.311 $\pm$0.018	**0.136** $\pm$0.019
FIGU + RIFF	0.155 $\pm$0.010	0.382 $\pm$0.039	0.104 $\pm$0.007

We repeat our setup using 5 different seeds for the model training and the sampling of the induction and selection sets. In Table 2 and 3 we report the average performance and average length of generated rule sets respectively, as

Table 3. Generated Rule set length for BAF, Credit and Industry Datasets

	BAF	**Industry**	**Taiwan Credit**
Expert Rules	-	13.0	-
CART + RIFF	10.4 $\pm$3.647	17.8 $\pm$4.087	5.2 $\pm$1.483
FIGS + RIFF	8.0 $\pm$1.732	9.2 $\pm$1.304	7.6 $\pm$1.949
FIGU + RIFF	**3.6** $\pm$0.548	**3.4** $\pm$0.548	**1.0** $\pm$0.000

well as the associated standard deviations. Interestingly, using RIFF on CART always improved its performance, a possible indication that CART was overfitting and RIFF reduced this by selecting its best rules. For the dataset with fewer samples, Taiwan Credit, RIFF increased the performance of CART and FIGS significantly, surpassing even LightGBM's performance. FIGU appears to generate rule sets that have similar performance to FIGS with much fewer rules, an indication that FIGU is able to reduce the overlap between generated trees, leading to shorter and, in theory, simpler to understand rule sets.

5 Conclusion And Future Work

In this work we propose RIFF, a rule induction algorithm that builds low FPR rule sets for fraud detection by greedily extracting rules from a tree based model like CART or FIGS. We also propose a slight modification to FIGS, FIGU, that aims to lower decision tree complexity so that it can be used by the RIFF selection algorithm to generate shorter rulesets. We perform a study with real world transaction data that shows that RIFF is able to perform better than expert rules, while maintaining the predictive performance of the original models and reducing their complexity.

While RIFF effectively generates a more concise and shorter rule set, it might provide complex, lengthier rules. We could expand the candidate set to also consider all nodes, instead of only leaves. This methodology draws a parallel to pruning methods, as this ideally leads RIFF into choosing more general, lower depth nodes in favour of their more specific, children nodes, similar to pruning.

A possible way to generate a more varied and robust rule set could involve extracting rules from all the trained CART and FIGS models into an unique candidate set. Since our setup subsamples the training set into subsests, and uses them to train these models, this is equivalent to applying the RIFF selection algorithm to a Random Forest [3] or Bagging FIGS [20].

References

1. Aparício, D., Barata, R., Bravo, J., Ascensão, J.T., Bizarro, P.: ARMS: automated rules management system for fraud detection. CoRR **abs/2002.06075** (2020)
2. Bache, K., Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
3. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (Oct 2001). https://doi.org/10.1023/A:1010933404324

4. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984), http://lyle.smu.edu/~mhd/8331f06/cart.pdf
5. Cendrowska, J.: Prism: An algorithm for inducing modular rules. International Journal of Man-Machine Studies **27**(4), 349–370 (1987). https://doi.org/10.1016/S0020-7373(87)80003-2
6. Clark, P., Niblett, T.: The cn2 induction algorithm. Machine Learning **3**, 261–283 (1989). https://doi.org/10.1023/A:1022641700528
7. Cohen, W.W.: Fast effective rule induction. In: Prieditis, A., Russell, S. (eds.) Machine Learning Proceedings 1995, pp. 115–123. Morgan Kaufmann, San Francisco (CA) (1995). https://doi.org/10.1016/B978-1-55860-377-6.50023-2
8. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001). https://doi.org/10.1214/aos/1013203451
9. Fürnkranz, J., Widmer, G.: Incremental reduced error pruning. In: ICML. pp. 70–77 (1994). https://doi.org/10.1016/B978-1-55860-335-6.50017-9
10. Hong J., Mozetic I., M.R.S.: Aq15: Incremental learning of attribute-based descriptions from examples, the method and user's guide. Reports of the Intelligent Systems Group (07 1986), https://hdl.handle.net/1920/1605
11. Iyer, R.K., Bilmes, J.A.: Submodular optimization with submodular cover and submodular knapsack constraints. CoRR **abs/1311.2106** (2013), http://arxiv.org/abs/1311.2106
12. Jesus, S., Pombal, J., Alves, D., Cruz, A., Saleiro, P., Ribeiro, R.P., Gama, J., Bizarro, P.: Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation. Advances in Neural Information Processing Systems (2022)
13. Kusters, R., Kim, Y., Collery, M., de Sainte Marie, C., Gupta, S.: Differentiable rule induction with learned relational features (2022). https://doi.org/10.48550/ARXIV.2201.06515
14. Michalski, R.S.: Pattern recognition as rule-guided inductive inference. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-2**, 349–361 (1980), https://api.semanticscholar.org/CorpusID:16719183
15. Michalski, R.S., Mozeti, I., Hong, J., Lavra, N.: The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In: AAAI Conference on Artificial Intelligence (1986), https://api.semanticscholar.org/CorpusID:18018701
16. Qiao, L., Wang, W., Lin, B.: Learning accurate and interpretable decision rule sets from neural networks. CoRR **abs/2103.02826** (2021), https://arxiv.org/abs/2103.02826
17. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**, 81–106 (1986). https://doi.org/10.1007/BF00116251
18. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning **5**, 239–266 (1990), https://api.semanticscholar.org/CorpusID:6746439
19. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993). https://doi.org/10.1007/BF00993309
20. Tan, Y.S., Singh, C., Nasseri, K., Agarwal, A., Yu, B.: Fast interpretable greedy-tree sums (FIGS). CoRR **abs/2201.11931** (2022)