LIBRE: Learning Interpretable Boolean Rule Ensembles

Graziano Mita **Paolo Papotti** **Maurizio Filippone** **Pietro Michiardi**
EURECOM, 06410 Biot (France)
{graziano.mita, paolo.papotti, maurizio.filippone, pietro.michiardi}@eurecom.fr

Abstract

We present a novel method – LIBRE – to learn an interpretable classifier, which materializes as a set of Boolean rules. LIBRE uses an ensemble of bottom-up, weak learners operating on a random subset of features, which allows for the learning of rules that generalize well on unseen data even in imbalanced settings. Weak learners are combined with a simple union so that the final ensemble is also interpretable. Experimental results indicate that LIBRE efficiently strikes the right balance between prediction accuracy, which is competitive with black-box methods, and interpretability, which is often superior to alternative methods from the literature.

1 INTRODUCTION

Model interpretability has become an important factor to consider when applying machine learning in critical application domains. In medicine, law, and predictive maintenance, to name a few, understanding the output of the model is at least as important as the output itself. However, a large fraction of models currently in use (e.g. DEEPNETS, SVMS) favor predictive performance at the expenses of interpretability.

To deal with this problem, interpretable models have flourished in the machine learning literature over the past years. Although defining interpretability is difficult (Miller, 2017; Doshi-Velez and Kim, 2017), the common goal of such methods is to provide an explanation of their output. The form and properties of the explanation are often application-specific.

In this work, we focus on predictive rule learning for challenging applications where data is imbalanced. For rules, interpretability translates into simplicity, and it is measured as a function of the number of rules and their size (average number of atoms): such proxies are easy to compute, understandable, and allow comparing several rule-based models. The goal is to learn a set of rules from the training set that (i) effectively predict a given target, (ii) generalize to unseen data, (iii) and are interpretable, i.e., a small number of short rules (e.g., fig. 1). The first objective is particularly difficult to meet in presence of imbalanced data. In this case, most rule-based methods fail at characterizing the minority class. Additional data issues that hinder the application of rule-based methods (Weiss, 2004) are data fragmentation (especially in case of *small-disjuncts* (Holte et al., 1989)), overlaps between imbalanced classes, and presence of rare examples.

IF mean_corpuscular_volume ∈ [90, 96)
OR gamma_glutamyl_transpeptidase ∈ [20, max]
THEN liver_disorder = True
ELSE liver_disorder = False

Figure 1: Example of rules learned by LIBRE for LIVER.

Many seminal rule learning methods come from the data mining community: CBA (Liu et al., 1998), CPAR (Yin and Han, 2003), and CMAR (Li et al., 2001), for example, use mining to identify class association rules and then choose a subset of them according to a ranking to implement the classifier. In practice, however, these methods output a huge number of rules, which negatively impacts interpretability.

Another family of approaches includes methods like CN2 (Clark and Niblett, 1989), FOIL (Quinlan and Cameron-Jones, 1993), and RIPPER-K (Cohen, 1995), whereby *top-down* learners build rules by greedily adding the condition that best explains the remaining data. *Top-down* learners are well suited for noisy data and are known to find general rules (Fürnkranz et al., 2014). They work well for the so called *large disjuncts*, but have difficulties to identify *small-disjuncts* and rare examples, which are quite common in imbalanced

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

settings. In contrast, *bottom-up* learners like MODLEM (Grzymala-busse and Stefanowski, 2001), start directly from very specific rules (the examples themselves) and generalize them until a given criteria is met. Such methods are susceptible to noise, and tend to induce a very high number of specific rules, but are better suited for cases where only few examples characterize the target class (Fürnkranz et al., 2014).

Hybrid approaches such as BRACID (Napierala, 2012) take the best from both worlds: *maximally-specific* (the examples themselves) and general rules are used together in a hybrid classification strategy that combines rule learning and instance-based learning. Thus, they achieve better generalization, also in imbalanced settings, but still generate many rules, penalizing interpretability. Other approaches to tackle data-related issues include heuristics to inflate the importance of rules for minority classes (Grzymala-Busse et al., 2000; Nguyen and Ho, 2005; Blaszczynski et al., 2010).

Recent work focus on marrying competitive predictive accuracy with *high interpretability*. A popular approach is to use the output of an association rule discovery algorithm (like FP GROWTH) and combine the discovered rules in a *small* and *compact* subset with high predictive performance. The rule combination process can be formalized either as an integer optimization problem or solved heuristically, explicitly encoding interpretability needs in the optimization function. Such approaches have been successfully applied to rule lists (Yang et al., 2017; Chen and Rudin, 2018; Angelino et al., 2018) and rule sets (Lakkaraju et al., 2016; Wang et al., 2017). Alternatively, rules can be directly learned from the data through an integer optimization framework (Hauser et al., 2010; Chang et al., 2012; Malioutov and Varshney, 2013; Goh and Rudin, 2014; Su et al., 2016; Dash et al., 2018).

Both rule-mining and integer-optimization based approaches underestimate the complexity and importance of finding good candidate rules, and become expensive when the input dimensionality increases, unless some constraints are imposed on the size and support of the rules. Although such constraints favour interpretability, they have a negative impact on the predictive performance of the model, as we show empirically in our work. Additionally, these methods do not consider class imbalance issues.

The key idea in our work is to exploit the known advantages of bottom-up learners in imbalanced settings, and improve their generalization and noise-tolerance through an ensembling technique that does not sacrifice interpretability. As a result, we produce a rule-based method that is (i) *versatile* and *effective* in dealing with both balanced and imbalanced data, (ii) *interpretable*, as it produces small and compact rule sets, and (iii) *scalable* to big datasets.

Contributions. (i) We propose LIBRE, a novel ensemble method that, unlike other ensemble proposals in the literature (W. Cohen and Singer, 1999; Friedman and Popescu, 2008; Dembczyński et al., 2010) is interpretable. Each weak learner uses a *bottom-up* approach based on monotone Boolean function synthesis and generates rules with no assumptions on their size and support. Candidate rules are then combined with a simple union, to obtain a final interpretable rule set. The idea of ensembling is crucial to improve generalization, while using *bottom-up* weak learners allows to generate meaningful rules even when the target class has few available samples. (ii) Our base algorithm for a weak learner, which is designed to generate a small number of compact rules, is inspired by Muselli and Quarati (2005), but it dramatically improves computational efficiency. (iii) We perform an extensive experimental validation indicating that LIBRE scales to large datasets, has competitive predictive performance compared to state-of-the-art approaches (even black-box models), and produces few and simple rules, often outperforming existing interpretable models.

2 BACKGROUND AND DEFINITIONS

Our methodology targets binary classification, although it can be easily extended to multi-class settings. For the sake of building interpretable models, we focus on Boolean functions for the mapping between inputs and labels, which are amenable to a simple interpretation.

Boolean functions can be used as a model for binary classifiers $f(\mathbf{x}) = y$, where $\mathbf{x} \in \{0,1\}^d$, $y \in \{0,1\}$. The function f induces a separation of $\{0,1\}^d$ in two subsets $\mathcal{F}$ and $\mathcal{T}$, where $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^d : f(\mathbf{x}) = 0\}$ and $\mathcal{T} = \{\mathbf{x} \in \{0,1\}^d : f(\mathbf{x}) = 1\}$. We call such subsets positive and negative subsets, respectively. Clearly, $\mathcal{F} \cup \mathcal{T} = \{0,1\}^d$ corresponds to the full truth table of the classification problem. We restrict the input space $\{0,1\}^d$ to be a *partially ordered set* (*poset*): a *Boolean lattice* on which we impose a partial ordering relation.

Definition 2.1. Let $\bigwedge$, $\bigvee$, $\neg$ be the AND, OR, and NOT logic operators respectively. A *Boolean lattice* is a 5 tuple $(\{0,1\}^d, \bigwedge, \bigvee, 0, 1)$. The lack of the $\neg$ operator implies that a lattice is **not** a Boolean algebra. Let $\leq$ be *a partial order relation* such that $\mathbf{x} \leq \mathbf{x}' \iff \mathbf{x} \bigvee \mathbf{x}' = \mathbf{x}'$. Then, $(\{0,1\}^d, \leq)$ is a poset, a set on which a partial order relation has been imposed.

The theory of Boolean algebra ensures that the class $\mathcal{B}_d$ of Boolean functions $f : \{0,1\}^d \to \{0,1\}$ can be re-

alized in terms of $\bigwedge$, $\bigvee$, and $\neg$. However, if $\{0,1\}^d$ is a Boolean lattice, $\neg$ is not allowed and only a subset $\mathcal{M}_d$ of $\mathcal{B}_d$ can be realized. The class $\mathcal{M}_d$ coincides with the collection of monotone Boolean functions. The lack of the $\neg$ operator may limit the family of functions we can reconstruct. However, by applying a suitable transformation of the input space, we can enforce the monotonicity constraint (Muselli, 2005). As a consequence, it is possible to find a function $\tilde{f} \in \mathcal{M}_d$ that approximates $f \in \mathcal{B}_d$ arbitrarly well.

Definition 2.2. Let $(\mathcal{X}, \leq)$ and $(\mathcal{Y}, \leq)$ be two posets. Then, $f : \mathcal{X} \to \mathcal{Y}$ is called *monotone* if $\mathbf{x} \leq \mathbf{x}'$ implies $f(\mathbf{x}) \leq f(\mathbf{x}')$.

Definition 2.3. Given $\mathbf{x} \in \{0,1\}^d$, let $\mathcal{I}_m$ be the set of the first m positive integers $\{1, \ldots, m\}$. $\mathcal{P}(\mathbf{x}) = \{i \in \mathcal{I}_m : \mathbf{x}(i) = 1\}$. The inverse of $\mathcal{P}$ is denoted as $p(\mathcal{P}(\mathbf{x}, m)) = \mathbf{x}$.

Definition 2.4. Let $\tilde{f} \in \mathcal{M}_d$ be a monotone Boolean function, and $\mathcal{A}$ be a partially ordered set. Then, $\tilde{f}$ can be written as: $\tilde{f}(\mathbf{x}) = \bigvee_{\mathbf{a} \in \mathcal{A}} \bigwedge_{j \in \mathcal{P}(\mathbf{a})} \mathbf{x}(j)$.

The monotone Boolean function $\tilde{f}$ is specified in *disjunctive normal form* (DNF), and is univocally determined by the set $\mathcal{A}$ and its elements. Thus, given $\mathcal{F}$ and $\mathcal{T}$, learning $\tilde{f}$ amounts to finding a particular set of lattice elements $\mathcal{A}$ defining the **boundary** separating positive from negative samples.

Definition 2.5. Given $\mathbf{a} \in \{0,1\}^d = \mathcal{T} \cup \mathcal{F}$, if $\mathbf{a} \leq \mathbf{x}$ for some $\mathbf{x} \in \mathcal{T}$, and $\nexists \mathbf{y} \in \mathcal{F} : \mathbf{a} \leq \mathbf{y}$, and $\exists \mathbf{y} \in \mathcal{F} : \mathbf{b} \leq \mathbf{y}, \forall \mathbf{b} < \mathbf{a}$, then $\mathbf{a}$ is a *boundary point* for $(\mathcal{T}, \mathcal{F})$. The set $\mathcal{A}$ of boundary points defines the *separation boundary*. If $\mathbf{a}' \not\leq \mathbf{a}''$ and $\mathbf{a}'' \not\leq \mathbf{a}', \forall \mathbf{a}', \mathbf{a}'' \in \mathcal{A}, \mathbf{a}' \neq \mathbf{a}''$, then the separation boundary is *irredundant*.

In other words, a boundary point is a lattice element that is smaller than or equal to at least one positive element in $\mathcal{T}$, but larger than all negative elements $\mathcal{F}$. In practical applications, however, we usually have access to a subset of the whole space, $\mathcal{D}_+ \subseteq \mathcal{T}$ and $\mathcal{D}_- \subseteq \mathcal{F}$. The goal of the algorithms we present next is to approximate the boundary $\mathcal{A}$, given $\mathcal{D}_+$ and $\mathcal{D}_-$. We show that boundary points, and binary samples in general, naturally translate into classification rules. Indeed, let $\mathcal{R}$ be the set of rules corresponding to the discovered boundary. $\mathcal{R}(\cdot)$ represents a binary classifier: $\mathcal{R}(\mathbf{x}) = \{1 \text{ if } \exists r \in \mathcal{R} : r(\mathbf{x}) = 1; 0 \text{ otherwise}\}$. Then, $\mathbf{x}$ is classified as positive if there is at least one rule in $\mathcal{R}$ that is true for it.

3 BOOLEAN RULE SETS

We presented a theoretical framework that casts binary classification as the problem of finding the boundary points for $\mathcal{D}_+ \subseteq \mathcal{T}$ and $\mathcal{D}_- \subseteq \mathcal{F}$. Next, we use such framework to design our interpretable classifier.

First, we describe a base, bottom-up method – which will be later used as a weak learner – that illustrates how to move inside the Boolean lattice to find boundary points. However, the base method does not scale to large datasets, and tends to overfit. Thus, we present LIBRE, an ensemble classifier that overcomes such limitations by running on randomly selected subset of features. LIBRE is interpretable because it combines the output of an ensemble of weak learners with a simple union operation. Finally, we present a procedure to select a subset of the generated points – the ones with the best predictive performance – and reduce the complexity of the boundary.

We assume that the input dataset is a poset and that the function we want to reconstruct is monotone. This is ensured by applying inverse-one-hot-encoding on discretized features, and concatenating the resulting binary features, as done in Muselli (2006). Given $z \in \mathcal{I}_m = \{1, ..., m\}$, inverse-on-hot encoding produces a binary string $\mathbf{b}$ of length m, where $b(i) = 1$ for $i \neq z$, $b(i) = 0$ for $i = z$. More details can be found in the supplementary material.

Example 3.1. Consider a dataset with two continuous features, f_1 and f_2, both taking values in the domain $[0, 100]$. Suppose that, a discretization algorithm outputs the following discretization ranges for the two features: $[[0, 40), [40, 100]]$ and $[[0, 30), [30, 60), [60, 100]]$ respectively. Once all records are discretized, we apply inverse one-hot encoding, as previously defined. For example, $f_1 = 33.1, f_2 = 44.7$ is first discretized as $f_1' = 1, f_2' = 3$, and then binarized as 01 101. In other words, each feature of a record is encoded with a number of bits equal to its discretized domain, and can have only one bit set to zero.

3.1 The Base, Bottom-up Method

We develop an approximate algorithm that learns the set $\mathcal{A}$ for $(\mathcal{D}_+, \mathcal{D}_-)$. The algorithm strives to find lattice elements such that both $|\mathcal{A}|$ and $|\mathcal{P}(\mathbf{a})|, \forall \mathbf{a} \in \mathcal{A}$ are small, translating in a small number of sparse boundary points (short rules).

Algorithm Design. To proceed with the presentation of our algorithm, we need the following definitions:

Definition 3.1. Given two lattice elements $\mathbf{x}, \mathbf{x}' \in \{0,1\}^d$, we say that $\mathbf{x}'$ *covers* $\mathbf{x}$, if and only if $\mathbf{x}' \leq \mathbf{x}$,

Definition 3.2. Given a lattice element $\mathbf{x} \in \{0,1\}^d$, *flipping off* the k-th element of $\mathbf{x}$ produces an element $\mathbf{z}$ such that $\mathbf{z}(i) = \mathbf{x}(i)$ for $i \neq k$ and $\mathbf{z}(i) = 0$ for $i = k$.

Definition 3.3. Given a positive binary sample $\mathbf{x} \in \mathcal{D}_+$, we say that a flip-off operation produces a *conflict* if the lattice element $\mathbf{z}$ resulting from the flip-off is such that $\exists \mathbf{x}' \in \mathcal{D}_- : \mathbf{z} \leq \mathbf{x}'$.

Then, a boundary point is a lattice element that covers at least one positive sample, and for which a flip-off operation would produce a conflict, as defined above.

Algorithm 1: FindBoundary

```
Set A = ∅ and S = D+;
while S ≠ ∅ do
    Choose x ∈ S;
    Set I = P(x), J = ∅;
    FindBoundaryPoint(A, I, J);
    Remove from S the elements covered by a, ∀a ∈ A;
end
```

Algorithm 1 presents the main steps of our algorithm, where $\mathcal{A}$ is the boundary set and $\mathcal{S} = \{\mathbf{s} \in \mathcal{D}_+ : \nexists \mathbf{a} \in \mathcal{A}, \mathbf{a} \leq \mathbf{s}\}$ is the set of elements in $\mathcal{D}_+$ that are not covered by a boundary point in $\mathcal{A}$. $\mathcal{I}$ is the set of indexes of the components of the current positive sample $\mathbf{x}$ that can be flipped-off, and $\mathcal{J}$ is the set of indexes that cannot be flipped-off to avoid a conflict with $\mathcal{D}_-$. Until $\mathcal{S}$ is not empty, an element $\mathbf{x}$ is picked from $\mathcal{S}$. Then, the procedure `FindBoundaryPoint` is used to generate one or more boundary points by flipping-off the candidate bits of $\mathbf{x}$. According to definition 3.2, a boundary point is generated when an additional flip-off would lead to a conflict, given definition 3.3. When the `FindBoundaryPoint` procedure completes its operation, both $\mathcal{A}$ and $\mathcal{S}$ are updated.

Example 3.2. Let $\mathcal{D}_+ = \{11001\}$ and $\mathcal{D}_- = \{01101, 01101\}$. Take the positive sample 11001, for which $\mathcal{I} = \{1, 2, 5\}$ and $\mathcal{J} = \emptyset$. Suppose that `FindBoundaryPoint` flips-off the bits in $\mathcal{I}$ from left to right. Flipping-off the first bit generates $01001 \leq 01101 \in \mathcal{D}_-$. The first bit is moved to $\mathcal{J}$ and kept to 1. Flipping-off the second bit generates $10001 \leq 10101 \in \mathcal{D}_-$. Also the second bit is moved to $\mathcal{J}$. We finally flip-off the last bit and obtain 11000 that is not in conflict with any element in $\mathcal{D}_-$. 11000 is therefore a boundary point for $(\mathcal{D}_+, \mathcal{D}_-)$.

If we think about binary samples in terms of rules, a positive sample can be seen as a maximally-specific rule, with equality conditions on the input features (the value that particular feature takes on that particular sample). Flipping-off bits is nothing more than generalizing that rule. Our goal is to do as many flip-off operations as possible before running into a conflict.

Retrieving the complete set of boundary points requires an exhaustive search, which is expensive, restricting its application to small, low-dimensional datasets. It is easy to show that the computational complexity of the exhaustive approach is $O(n^2 2^d)$, where n is the number of *distinct* training samples, and d is the dimension of the Boolean lattice. In this work, we propose an *approximate heuristic* for the `FindBoundaryPoint` procedure.

Finding Boundary Points. The key idea is to find a subset of all possible boundary points, steering their selection through a measure of their quality. A boundary point is considered to be "good" if it contributes to decreasing the complexity of the resulting boundary set, which is measured in terms of its cardinality $|\mathcal{A}|$ and the total number of positive bits $\sum_{\mathbf{a}\in\mathcal{A}} |\mathcal{P}(\mathbf{a})|$. In practice, $|\mathcal{A}|$ can be decreased by choosing boundary points that cover the largest number of elements in $\mathcal{S}$. To do this, we iteratively select the best candidate index $i \in \mathcal{I}$ according to a measure of potential coverage. Decreasing $\sum_{\mathbf{a}\in\mathcal{A}} |\mathcal{P}(\mathbf{a})|$ implies finding boundary points with low number of 1s.

Before proceeding, we define a notion of distance between lattice elements:

Definition 3.4. Given $\mathbf{x}, \mathbf{x}' \in \{0,1\}^d$, the *distance* $d_l(\mathbf{x}, \mathbf{x}')$ between $\mathbf{x}$ and $\mathbf{x}'$ is defined as: $d_l(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} |\mathbf{x}(i) - \mathbf{x}'(i)|_+$, where $|\cdot|_+$ is equal to 1 if $(\cdot) \geq 0$, 0 otherwise.

Definition 3.5. In the same way, we can define the distance between a lattice element $\mathbf{x}$ and a set $\mathcal{V}$ as: $d_l(\mathbf{x}, \mathcal{V}) = \min_{\mathbf{x}'\in\mathcal{V}} d_l(\mathbf{x}, \mathbf{x}')$.

Every boundary point $\mathbf{a}$ for $(\mathcal{D}_+, \mathcal{D}_-)$ has distance $d_l(\mathbf{a}, \mathcal{D}_-) = 1$; in fact, boundary points are all lattice elements for which a flip-off would generate a conflict. In the iterative selection process of the best index $i \in \mathcal{I}$ to be flipped-off, indexes having high $d_l(p(\mathcal{I}\cup\mathcal{J}), \mathcal{D}_-{}_i^0)$ are preferred, where $\mathcal{D}_-{}_i^0 = \{\mathbf{x} \in \mathcal{D}_- : \mathbf{x}(i) = 0\}$, because they are the ones that contribute most to reduce the number of 1s of a potential boundary point.

Algorithm 2: FindBoundaryPoint($\mathcal{A}, \mathcal{I}, \mathcal{J}$)

```
For each i ∈ I compute |S_i^0|, |D+_i^0|, d_l(p(I ∪ J), D-_i^0);
while I ≠ ∅ do
    Move from I to J all i with d_l(p(I ∪ J), D-_i^0) = 1;
    if I = ∅ then
        break;
    end
    Choose the best index i ∈ I;
    Remove i from I;
    For each i ∈ I update d_l(p(I ∪ J), D-_i^0);
end
if there is no a ∈ A : p(J) ≥ a then
    Set A = A ∪ p(J);
end
```

Algorithm 2 illustrates our approximate procedure, where $\mathcal{S}_i^0 = \{\mathbf{s} \in \mathcal{S} : \mathbf{s}(i) = 0\}$ and $\mathcal{D}_+{}_i^0 = \{\mathbf{t} \in \mathcal{D}_+ : \mathbf{t}(i) = 0\}$ are proxies for the potential coverage of flipping-off a given bit i. The first step of the algorithm computes, for each index $i \in \mathcal{I}$, the terms $|\mathcal{S}_i^0|$ and $|\mathcal{D}_+{}_i^0|$ indicating its potential coverage, and $d_l(p(\mathcal{I} \cup \mathcal{J})$. Until the set $\mathcal{I}$ is not empty, indexes inducing a unit distance to $\mathcal{D}_-$ are moved to $\mathcal{J}$. Then, we choose the best index i_{best} among the remaining indices in $\mathcal{I}$, using our **greedy heuristics**: we can chose to optimize either for the tuple

$H_1 = (|\mathcal{S}_i^0|, |\mathcal{D}_{+i}^0|, d_l(p(\mathcal{I} \cup \mathcal{J}), \mathcal{D}_{-i}^0))$ or for the tuple $H_2 = (d_l(p(\mathcal{I} \cup \mathcal{J}), \mathcal{D}_{-i}^0), |\mathcal{S}_i^0|, |\mathcal{D}_{+i}^0|)$. H_1 prioritizes a lower number of boundary points, while H_2 tends to generate boundary points with fewer 1s.

When $\mathcal{I}$ is empty, $p(\mathcal{J})$ is added to the boundary set $\mathcal{A}$ if it does not contain already an element covering $p(\mathcal{J})$. Note that, in algorithm 2, the distance is computed only once, and updated at each iteration. This is because only one bit is selected and removed from $\mathcal{I}$; then, $p(\mathcal{I} \cup \mathcal{J})_{new} = p((\mathcal{I} \cup \mathcal{J})_{old} \setminus \{i\})$. Formally, we apply definition 3.4 exclusively for $i = i_{best}$.

Example 3.3. Let $\mathcal{D}_+ = \{10101, 01101, 01110\}$ and $\mathcal{D}_- = \{10110, 11010\}$. We describe the procedure for few steps and only for the first positive sample 10101. Suppose to optimize the tuple $(|\mathcal{S}_i^0|, |\mathcal{D}_{+i}^0|, d_l(p(\mathcal{I} \cup \mathcal{J})))$. For 10101 we have $\mathcal{I} = \{1, 3, 5\}$ and $\mathcal{J} = \emptyset$. At the beginning $\mathcal{S} = \mathcal{D}_+$. $|\mathcal{D}_{+1}^0| = 2, |\mathcal{D}_{+3}^0| = 0, |\mathcal{D}_{+5}^0| = 1$. $\mathcal{D}_{-1}^0 = \emptyset, \mathcal{D}_{-3}^0 = \{11010\}, \mathcal{D}_{-5}^0 = \{10110, 11010\}$. Consequently: $d_l(p(\mathcal{I} \cup \mathcal{J}), \mathcal{D}_{-1}^0) = undefined$, $d_l(p(\mathcal{I} \cup \mathcal{J}), \mathcal{D}_{-3}^0) = 2$, $d_l(p(\mathcal{I} \cup \mathcal{J}), \mathcal{D}_{-5}^0) = 1$. Bit 5 is moved to $\mathcal{J}$. Bit 1 has the higher value of $|\mathcal{D}_{+i}^0|$ and is selected as best candidate to be flipped-off. The distance is recalculated and the procedure continues until the set of candidate bits $\mathcal{I}$ is empty.

The algorithmic complexity of algorithm 1, when it runs algorithm 2, is $O(n^2 d^2)$. This is faster than the exhaustive algorithm, and better than the $O(n^2 d^3)$ complexity of Muselli and Quarati (2005). We refer the reader to the supplement for additional details on the difference between our proposal and Muselli and Quarati (2005). We also point out that most sequential-covering algorithms repeatedly remove the samples covered by the new rules, forcing the induction phase to work in a more partitioned space with less data, especially affecting minority rules, which already rely on few samples. The problem is mitigated in our solution: despite $\mathcal{S}$ cannot avoid this behavior, our heuristics keep a global and constant view of both $\mathcal{D}_-$, in the conflict detection, and $\mathcal{D}_+$, in the discrimination of the best bits to flip.

From Boundary Set To Rules. Each element $\mathbf{a}$ of the boundary set $\mathcal{A}$ can be practically seen as the antecedent of an **if-then** rule having as target the positive class. When a binary sample $\mathbf{x}$ is presented to $\mathbf{a}$, the rule outputs 1 only if $\mathbf{x}$ has a 1 in all positions where $\mathbf{a}$ has value 1, that is if $\mathbf{a} \leq \mathbf{x}$. Then, the antecedent of the rule is expressed as a function of the input features in the original domain.

Example 3.4. Consider a dataset with two continuous features, f_1 and f_2, discretized as follows: $[[0, 40), [40, 100]]$ and $[[0, 30), [30, 60), [60, 100]]$ respectively. Let's assume that our algorithm outputs a boundary set $\mathcal{A} = \{01\ 100\}$. From the boundary point we obtain a rule as follows: the first two bits referring to feature f_1 – 01 – are mapped to "**if** $f_1 \in [0, 40)$", while the bits referring to f_2 – 100 – are mapped to "**if** $f_2 \in [30, 100]$", where the two consecutive intervals have been combined. The zeros determine the ranges in the if conditions. The final rule is therefore "**if** $f_1 \in [0, 40)$ and $f_2 \in [30, 100]$ **then** label = 1".

3.2 The LIBRE Method

The base approach generates boundary points by generalizing input samples, i.e., by flipping-off positive bits if no conflict with negative samples is encountered. The hypothesis underlying this procedure is that when no conflicts are found, a boundary point induces a valid rule. However, such rule might be violated when used with unseen data. Stopping the flipping-off procedure as soon as a single conflict is found has two main effects: i) we obtain very specific rules, that might be simplified if the approach could tolerate a limited number of conflicts; ii) the rules cover no negative samples in the training set and tend to overfit.

To address these issues, a simple method would be to introduce a measure for the number of conflicts and use it as an additional heuristic in the learning process. However, this would dramatically increase the complexity of the algorithm.

A more natural way to overcome such challenges is to make the algorithm directly work on (random) subsets of features; in this way, the learning process produces more general rules by construction. Randomization is a well-known technique to implement ensemble methods that provide superior classification accuracy, as demonstrated, for example, in random forests (Ho, 1998; Breiman, 2001). By using randomization, we can directly use the methodology described in the previous sections, without modifying the search procedure. The new approach – LIBRE – is an interpretable ensemble of rules that operates on a randomized subset of features.

Formally, let E be the number of classifiers in the ensemble. For each classifier $j \in \{1, \ldots, E\}$, we randomly sample k_j features of the original space and run algorithm 1 to produce a boundary set $\mathcal{A}_j$ for the reduced input space. $\mathcal{A}_j$ can be generated in parallel, since weak learners are independent from each other. At this point, to make the ensemble interpretable, we crucially do not apply a voting (or aggregation) mechanism to produce the final class prediction, but we do a simple union, such that $\mathcal{A} = \bigcup_{j=1}^{E} \mathcal{A}_j$.

We note that LIBRE addresses the problems outlined above, as we show experimentally. By training an ensemble of *weak learners* that operate on a small subset of features, we artificially inflate the probability of finding negative examples. Each weak learner is

constrained to run on less features not only reducing the impact of d on the execution time, but also having an immediate effect on the interpretability of the model that is forced to generate simpler rules, exactly because it operates on fewer input features.

Note that there are no guarantees that elements of $\mathcal{A}_j$ will actually be boundary points in the full feature space: weak learners have only a partial view of the full input space and might generate rules that are not globally true. Thus, it is important to filter out the points that are clearly far from the boundary by using the selection procedure described in the next section.

3.3 Producing The Final Boundary

The model learned by our greedy heuristic materializes as a set $\mathcal{A}$, which might contain a large number of elements and, in case of LIBRE, it might also contain elements that cover many negative samples. In this section, we explain how to produce a boundary set $\mathcal{A}^*$ with a good tradeoff between complexity and predictive performance. This can be cast as a *weighted set cover* problem. Since exploring all possible subsets of elements in $\mathcal{A}$ can be computationally demanding, we use a standard greedy weighted set cover algorithm.

Each element $\mathbf{a} \in \mathcal{A}$ is intially assigned a weight $w(\mathbf{a}) = \alpha|\mathcal{P}(\mathbf{a})| - (1-\alpha)|\mathcal{N}(\mathbf{a})|$ that is proportional to the number of positive and negative covered samples, $|\mathcal{P}(\mathbf{a})|$ and $|\mathcal{N}(\mathbf{a})|$ respectively. The importance of the two contributions is governed by a parameter α. At each iteration, the element $\mathbf{a}$ with the highest weight is selected; if there is more than one, the element with the highest number of zeros is preferred. All samples that are covered by the selected element are removed, and the weights are recalculated. The process continues until either all samples are a covered or a stopping condition is met. In imbalanced settings, α will be close to 1 to increase the strength of minority rules.

Before running the selection procedure, with the aim of speeding up execution times, we eventually apply a filtering procedure to reduce the size of the initial set to a small number of good candidates: as proposed by Gu et al. (2003), we select the top K rules according to *exclusiveness* and *local support*, that are more sensible than confidence and support for imbalanced settings.

4 EXPERIMENTS

We evaluate LIBRE in terms of predictive performance, interpretability, and scalability, and compare it with other rule-based and black-box methods.

Datasets. We report the results for seven publicly available datasets from the UCI repository and two real industrial IT datasets – proprietary of SAP. Results on other UCI datasets are in the supplementary material. These datasets cover several domains, have different imbalance ratios, number of records and features, as summarized in Table 1. Some of these datasets have been used to evaluate methods for class imbalance (Van Hulse et al., 2007) and present characteristics that make them difficult to learn: overlapping classes, noisy and rare examples. The SAP datasets consist of monitoring data collected across database systems. They have 45 features, hand-crafted by domain experts based on low-level system metrics. SAP runs a predictive maintenance system on this data and notifies customers who confirm or discard the warnings: we use these as binary labels. SAP-CLEAN is the clean version of SAP-FULL, where we removed records with at least one missing value. We refer the reader to the supplement for details on data preprocessing.

Dataset	#records	#features	imbalance_ratio
ADULT	48'842	14	.23
AUSTRALIAN	690	14	.44
BANK	45'211	17	.12
ILPD	583	10	.28
LIVER	345	5	.51
PIMA	768	8	.35
TRANSFUSION	748	5	.24
SAP-CLEAN	287'031	45	.01
SAP-FULL	1'554'227	45	.01

Table 1: Characteristics of evaluated datasets.

Comparison With Other Methods. We compare LIBRE with two recent works: Scalable Bayesian Rule Lists (S-BRL) (Yang et al., 2017) and Bayesian Rule Sets (BRS) (Wang et al., 2017). We also report the results for a WEKA implementation of RIPPER-K (Cohen, 1995) and MODLEM (Grzymala-busse and Stefanowski, 2001) – as representative of top-down and bottom-up approaches – and SCIKIT-LEARN implementations of Decision Tree (DT) (Breiman et al., 1984), Support Vector Machine with RBF kernel (RBF-SVM) (Cortes and Vapnik, 1995)), and random forests (RF) (Breiman, 2001). RBF-SVM and RF are selected as popular black-box models; RF is also a representative ensemble method. Other relevant methods are not publicly available (CG (Dash et al., 2018)), do not work properly (IDS (Lakkaraju et al., 2016)), or are only partially implemented (BRACID (Napierala, 2012)).

Parameter Tuning. The initial set of candidate rules for S-BRL and BRS is generated by running FP GROWTH with a minimum support of 1 and a maximum mining length of 5. We also optimize BRS and S-BRL's prior hyperparameters by cross validation. For BRS, we run 2 chains of 500 iterations. For RIPPER-K, we change the number of optimization steps be-

Dataset	RBF-SVM	RF	DT	RIPPER-K	MODLEM	S-BRL	BRS	LIBRE	LIBRE 3
ADULT	.62(.01)	.68(.01)	.68(.01)	.59(.02)	.66(.01)	.68(.01)	.61(.01)	**.70(.01)**	.62(.01)
AUSTRALIAN	.83(.02)	**.86(.02)**	.84(.02)	.85(.02)	.68(.28)	.82(.03)	.83(.03)	.84(.03)	.84(.03)
BANK	.46(.01)	.50(.01)	.50(.01)	.44(.04)	.50(.03)	.50(.02)	.32(.05)	**.55(.01)**	.44(.01)
ILPD	.47(.02)	.44(.08)	.42(.10)	.20(.11)	.48(.08)	.14(.13)	.09(.08)	**.54(.06)**	.52(.04)
LIVER	.58(.08)	.58(.07)	.56(.10)	.59(.04)	.58(.07)	.54(.03)	.61(.05)	.60(.07)	**.63(.06)**
PIMA	.61(.04)	.63(.04)	.60(.01)	.60(.03)	.38(.18)	.61(.07)	.03(.03)	**.64(.05)**	**.64(.05)**
TRANSFUSION	.41(.07)	.35(.06)	.35(.05)	.42(.10)	.42(.08)	.05(.10)	.04(.05)	**.49(.12)**	**.49(.12)**
SAP-CLEAN	.93(.02)	.93(.01)	.85(.03)	.86(.02)	.88(.01)	.90(.01)	.68(.03)	**.95(.02)**	.72(.03)
SAP-FULL	-	-	-	-	-	.81(.02)	-	**.89(.03)**	.68(.04)
Avg Rank	4.7(1.2)	3.3(1.6)	4.9(2.1)	5.3(2.1)	4.9(2.2)	5.2(2.8)	7.2(2.5)	**1.4(0.8)**	3.6(2.6)

Table 2: F1-score (st. dev. in parenthesis).

Dataset	DT	RIPPER-K	MODLEM	S-BRL	BRS	LIBRE	LIBRE 3
ADULT	287.8(6.5)	21.4(5.2)	4957.8(36.3)	71.4(2.1)	10.0(3.3)	14.0(2.1)	**3.0(0.0)**
AUSTRALIAN	4.0(0.0)	3.8(1.2)	86.6(3.2)	5.8(0.7)	**1.8(0.4)**	2.4(1.4)	2.2(0.7)
BANK	545.4(18.3)	9.0(1.8)	3722.6(25.5)	61.2(5.5)	4.8(1.2)	15.0(1.1)	**2.0(0.6)**
ILPD	80.6(30.2)	**1.0(0.6)**	128.2(7.8)	4.8(0.7)	**1.0(0.0)**	4.4(2.3)	2.2(0.4)
LIVER	84.4(15.2)	1.4(0.8)	98.4(1.6)	4.0(0.6)	**1.0(0.0)**	3.4(1.9)	2.8(0.4)
PIMA	84.8(43.1)	2.4(2.4)	151.8(7.6)	8.4(0.5)	**1.0(0.0)**	1.6(1.0)	1.6(1.0)
TRANSFUSION	100.2(48.4)	1.8(0.4)	125.8(6.1)	4.4(0.8)	**1.0(0.0)**	1.2(0.4)	1.2(0.4)
SAP-CLEAN	622.4(51.9)	19.3(3.6)	3944.5(18.8)	47.7(4.4)	20.2(3.5)	13.0(2.4)	**3.0(0.0)**
SAP-FULL	-	-	-	56.4(4.6)	-	17.5(5.2)	**3.0(0.0)**
Avg Rank	5.7(0.7)	3.2(1.0)	6.7(0.9)	4.9(0.7)	1.9(1.2)	2.9(0.9)	**1.8(0.8)**

Table 3: #rules (st. dev. in parenthesis).

tween 1 and 5, and activate pruning. For MODLEM, we try all available classification strategies and condition measures. For RBF-SVM, we optimize C and γ. For DT and RF, we optimize the maximum depth in $\{5, 10, 20, None\}$, we tried all possible options for max_features and use a number of trees in $\{20, 50, 100\}$ for RF. For LIBRE, we vary the number of weak learners in $E \in \{5, 20, 50\}$. Each weak learner uses up to 5 features. Additionally, we try the two heuristics H_1 and H_2 to generate rules and vary α in $\{.5, .7, .9\}$ for weighted set cover. Parameters not reported above are all fixed to recommended or default values.

Evaluation Metrics. All results refer to nested 5-fold cross validation, where the same splits are used for all methods. We use F1-score to compare the predictive performance of the classifiers, as it is well-suited to evaluate the capability to characterize the target class both in balanced and imbalanced settings. For rule-based methods, we use standard metrics from the literature to evaluate the interpretability of the rule sets, namely the *number of rules* that implement a model, and the *average number of atoms per rule*. For DT, we extract the rules following the paths from root to leaves: this captures the perception of a user who looks at the tree to understand the output of the model. For S-BRL, the number of atoms in a rule is equal to the sum of the atoms in the previous rules, highlighting the fact that a user has to go through all the rules up to the one that returns the label. For all rule-based methods, we change inequalities $(<, \leq, >, \geq)$ to ranges to have a fair comparison. For example, $f_1 \geq 3$ is converted to $f_1 \in [3, max]$.

Predictive Performance Evaluation. Table 2 shows the means and standard deviations of the F1-score for the tested algorithms (best results in bold) and the rank of their average performance, where the same splits are used for all tested methods. We additionally report the results for LIBRE when it is constrained to generate at most 3 rules (LIBRE 3).

If we look at the average rank, LIBRE emerges as the best method, beating both RBF-SVM and RF, demonstrating its versatility in both balanced and imbalanced settings. LIBRE 3 is still better than the other rule-based competitors, although being constrained to generate at most 3 rules. DT, MODLEM, S-BRL and RIPPER-K show very similar performance, even if MODLEM is usually worse for balanced settings. BRS is the worst method in terms of predictive performance.

Focusing more on the single datasets, we can see that, except for AUSTRALIAN, LIBRE obtains consistently the highest F1-score. In BANK, ILPD, and TRANSFUSION the gap between LIBRE and the closest competitor is significant; the gap is even larger in comparison to alternative rule-based methods. For the remaining datasets, the differences with the competitors are less pronounced but still significant. In particular, ILPD seems to be very problematic for most of the tested methods: RIPPER-K, BRS and S-BRL do not learn anything useful about the positive class; MODLEM per-

forms marginally better. From a deeper analysis, it emerges that ILPD is an imbalanced dataset with overlapping classes: rules learned by LIBRE have an error rate close to 50% on the training set, consequence of the class imbalance. RIPPER-K is not able to learn these rules, whereas the selection stage of BRS and S-BRL does not include such rules in the final set even when they are in the set of candidate mined rules.

With SAP-CLEAN, LIBRE 3 performs better than BRS but limiting the number of rules to 3 causes a significant drop in F1-score w.r.t. LIBRE. The situation is different for SAP-FULL, the original version of the dataset containing also missing values. From table 1, SAP-FULL is more than five times bigger than SAP-CLEAN, indicating that missing values are not a negligible problem in real scenarios. A method that runs without additional preprocessing is thus truly desirable. Only LIBRE and S-BRL fit this requirement, while RIPPER-K, BRS, and MODLEM natively manage missing values for categorical features only, but require an additional preprocessing for continuous features. Despite the huge number of missing values, results for LIBRE are comparable to other rule-based methods when executed on SAP-CLEAN.

Interpretability Evaluation. Next, using table 3, we evaluate interpretability in terms of quantity and simplicity of rules. In our analysis, we also refer to table 2, to measure the trade-off that exists between interpretability and predictive performance. We highlight in bold the most interpretable results.

In terms of number of rules, LIBRE is better than RIPPER-K on average, indicating that it indeed overcomes the limitations of bottom-up learners like MODLEM, that is instead the worst method together with DT. S-BRL is competitive for small datasets, but the number of rules increases considerably for bigger datasets like ADULT, BANK, and SAP. Overall, BRS generates compact rule sets, with only one rule for half of the tested datasets. However, we should also notice that, except for LIVER, these are the same datasets that give F1-score close to zero. LIBRE 3 outperforms other methods and produces the most compact rule sets for the three larger datasets, with a small impact on predictive performance.

All the tested methods have a similar average number of atoms(results in the supplementary material). Only S-BRL has issues when the number of rules is significant (like in ADULT, BANK, and SAP datasets): indeed, in rule lists every rule depends on the previous ones, and the number of atoms easily explodes.

Scalability Evaluation. Table 4 shows the run time for LIBRE and three representative rule-based competitors on synthetic balanced datasets with 10 features and a varying number of records: from 10'000 to 1'000'000. For each configuration, we randomly generate the dataset 3 times and report the average run time and standard deviation. All methods are tested with their default parameters and run sequentially, for a fair comparison. For LIBRE, the time refers to one weak learner, which is also a good approximation for the computing time of E parallel weak learners. The symbol "-" identifies *out-of-memory* errors.

#records	RIPPER-K	MODLEM	BRS	LIBRE
10'000	1(0)	14(0)	144(1)	5(0)
100'000	7(3)	2457(89)	2994(304)	44(5)
500'000	39(25)	-	-	209(7)
1'000'000	101(31)	-	-	399(8)

Table 4: Runtime in seconds (st. dev. in parenthesis).

MODLEM and BRS fail with an out-of-memory error with 500'000 and 1'000'000 records datasets. They also show much higher run times for smaller datasets w.r.t. RIPPER-K and LIBRE, that are instead able to complete their training in a few minutes also for the large datasets. Note that each weak learner in LIBRE works with $\mathcal{D}_+$ and $\mathcal{D}_-$ that consist of *distinct records*: even if the original dataset has millions of entries, the number of binary records processed by the algorithm is much lower, especially when the number of input features of each weak learner is relatively low. We also point out that, for practical applications where interpretability is needed, it is convenient to limit the number of features and train a bigger ensemble with more learners to quickly generate understandable rules.

5 CONCLUSION

Model interpretability has recently become of primary importance in many applications. In this work, we focused on the task of learning a set of rules which specify, using Boolean expressions, the classification model. We devised a practical method based on monotone Boolean function synthesis to learn rules from data. Our approach uses an ensemble of bottom-up learners that generalizes better than traditional bottom-up methods, and that works well for both balanced and imbalanced scenarios. Interpretability needs can be easily encoded in the rule generation and selection procedure that produces short and compact rule sets.

Our experiments show that LIBRE strikes the right balance between predictive performance and interpretability, often outperforming alternative approaches from the literature. For future work, we will extend our model considering noisy labels and a Bayesian formulation.

Acknowledgements

The authors thank SAP Labs France for their support. MF gratefully acknowledges support from the AXA Research Fund.

References

E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. Learning certifiably optimal rule lists for categorical data. *JMLR*, 18(234):1–78, 2018.

J. Blaszczynski, M. Deckert, J. Stefanowski, and S. Wilk. Integrating selective pre-processing of imbalanced data with ivotes ensemble. In *Proc. of the 7th Int. Conf. on Rough Sets and Current Trends in Computing, RSCTC*, pages 148–157, 2010.

L. Breiman. Random forests. *Mach. Learn.*, 45(1): 5–32, 2001. ISSN 0885-6125.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

A. Chang, D. Bertsimas, and C. Rudin. An integer optimization approach to associative classification. In *Proc. of the 24th Int. Conf. on Neural Information Processing Systems, NIPS*, pages 3302–3310, 01 2012.

C. Chen and C. Rudin. An optimization approach to learning falling rule lists. In *Proc. of the 21st Int. Conf. on Artif. Intel. and Stat., AISTATS*, pages 604–612, 2018.

P. Clark and T. Niblett. The cn2 induction algorithm. *Mach. Learn.*, 3(4):261–283, 1989. ISSN 1573-0565.

W. W. Cohen. Fast effective rule induction. In *Proc. of the 20th Int. Conf. on Mach. Learn., ICML*, pages 115–123, 1995.

C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

S. Dash, O. Günlük, and D. Wei. Boolean decision rules via column generation. In *Proc. of the 31st Int. Conf. on Neural Information Processing Systems, NIPS*, 2018.

K. Dembczyński, W. Kotłowski, and R. Słowiński. Ender: a statistical framework for boosting decision rules. *Data Min. and Knowl. Disc.*, 21(1):52–90, 2010. ISSN 1573-756X.

F. Doshi-Velez and B. Kim. A roadmap for a rigorous science of interpretability. *CoRR in arXiv*, 2017.

J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Appl. Stat.*, 2(3): 916–954, 2008.

J. Fürnkranz, D. Gamberger, and N. Lavrač. *Foundations of Rule Learning*. Springer Publishing Company, Incorporated, 2014.

S. T. Goh and C. Rudin. Box drawings for learning with imbalanced data. In *Proc. of the 20th Int. Conf. on Knowl. Disc. and Data Min., KDD*, pages 333–342, 2014.

J. W. Grzymala-busse and J. Stefanowski. Three discretization methods for rule induction. *Int. Journal of Intelligent Systems*, pages 29–38, 2001.

J. W. Grzymala-Busse, L. K. Goodwin, W. J. Grzymala-Busse, and X. Zheng. An approach to imbalanced data sets based on changing rule strength. In *Rough-Neural Computing: Techniques for Computing with Words*, 2000.

L. Gu, J. Li, H. He, G. Williams, S. Hawkins, and C. Kelman. Association rule discovery with unbalanced class distributions. In *Proc. of the 16th Austr. Conf. on Artif. Intel., AI*, pages 221–232, 12 2003.

J. R. Hauser, O. Toubia, T. Evgeniou, R. Befurt, and D. Dzyabura. Disjunctions of conjunctions, cognitive simplicity, and consideration sets. *Journal of Marketing Res.*, 47(3):485–496, 2010.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intel., TPAMI*, 20(8):832–844, 1998. ISSN 0162-8828.

R. C. Holte, L. E. Acker, and B. W. Porter. Concept learning and the problem of small disjuncts. In *Proc. of the 11th Int. Joint Conf. on Artif. Intel., IJCAII*, pages 813–818, 1989.

H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proc. of the 22nd Int. Conf. on Knowl. Disc. and Data Min., KDD*, pages 1675–1684, 2016.

W. Li, J. Han, and J. Pei. Cmar: accurate and efficient classification based on multiple class-association rules. In *Proc. of the 2001 IEEE Int. Conf. on Data Min., ICDM*, pages 369–376, 2001.

B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of the 4th Int. Conf. on Knowl. Disc. and Data Min., KDD*, pages 80–86, 1998.

D. Malioutov and K. Varshney. Exact rule learning via boolean compressed sensing. In *Proc. of the 30th Int. Conf. on Mach. Learn., ICML*, pages 765–773, 2013.

T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *CoRR in arXiv*, 2017.

M. Muselli. Approximation properties of positive boolean functions. In *Proc. of the 16th WIRN/NAIS*, 2005.

M. Muselli. Switching neural networks: A new connectionist model for classification. In B. Apolloni,

M. Marinaro, G. Nicosia, and R. Tagliaferri, editors, *Neural Nets*, pages 23–30, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

M. Muselli and A. Quarati. Reconstructing positive boolean functions with shadow clustering. In *Proc. of the 2005 Europ. Conf. on Circuit Theory and Design, ECCTD*, volume 3, pages III/377 – III/380 vol. 3, 2005.

K. Napierala. Bracid: a comprehensive approach to learning rules from imbalanced data. *Journal of Intel. Information Systems*, 39(2):335–373, 2012.

C. H. Nguyen and T. Ho. An imbalanced data rule learner. In *Proc. of the 9th Europ. Conf. on Principles and Practice of Knowl. Disc. in Databases, PKDD*, volume 3721, pages 617–624, 10 2005.

J. R. Quinlan and R. M. Cameron-Jones. Foil: A midterm report. In *Proc. of the 4th Europ. Conf. on Mach. Learn., ECML*, pages 1–20, 1993.

G. Su, D. Wei, K. R. Varshney, and D. M. Malioutov. Learning sparse two-level boolean rules. In *Proc. of the IEEE 26th Int. Workshop on Mach. Learn. for Signal Processing, MLSP*, pages 1–6, 2016.

J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano. Experimental perspectives on learning from imbalanced data. In *Proc. of the 24th Int. Conf. on Mach. Learn., ICML*, ICML '07, pages 935–942, 2007.

W. W. Cohen and Y. Singer. A simple, fast, and effective rule learner. *Proc. of the 16th Nat. Conf. on Artif. Intel., AAAI*, 06 1999.

T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. A bayesian framework for learning rule sets for interpretable classification. *JMLR*, 18(70):1–37, 2017.

G. M. Weiss. Mining with rarity: A unifying framework. *SIGKDD Explorations*, 6(1):7–19, June 2004.

H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *Proc. of the 34th Int. Conf. on Mach. Learn., ICML*, pages 3921–3930, 2017.

X. Yin and J. Han. *CPAR: Classification based on Predictive Association Rules*, pages 331–335. SIAM, 2003.