Optimizing Binary Decision Diagrams for Interpretable Machine Learning Classification

Optimizing Binary Decision Diagrams for Interpretable Machine Learning Classification

Gianpiero Cabodi*, Paolo E. Camurati*, Joao Marques-Silva†, Marco Palena* and Paolo Pasini*
*DAUIN, *Politecnico di Torino*, Turin, IT
{gianpiero.cabodi,paolo.camurati,marco.palena,paolo.pasini}@polito.it
†ANITI, *University of Toulouse*, Toulouse, FR
joao.marques-silva@univ-toulouse.fr

***Abstract*—Machine learning (ML) is ever more frequently used as a tool to aid decision-making. The need to understand the decisions made by ML algorithms has sparked a renewed interest in explainable ML models. A number of known models are often regarded as interpretable by human decision-makers with varying degrees of difficulty. The size of such models plays a crucial role in determining how easily they can be understood by a human. In this paper[1] we propose the use of Binary Decision Diagrams (BDDs) as an interpretable ML model. BDDs can be deemed as interpretable as decision trees (DTs) while offering a often more compact representation due to node sharing. Fixed variable ordering also allows for more concise explanations. We propose a SAT-based approach for learning optimal BDDs that exhibit perfect accuracy on training data. We also explore heuristic methods for computing sub-optimal BDDs, in order to improve scalability.**

I. Introduction

The increasing use of Machine Learning (ML) to automate decision-making motivates the need to explain the outcome of ML algorithms to the general public. The importance of this "right to explanation" has been recognized in recent legislative efforts [2]. Social acceptance of ML-made decisions entails going beyond the classical black-box model. The area of *eXplainable Artificial Intelligence* (XAI) [3] responds to such a need by providing models whose outcomes can be easily explained to human decision-makers.

In this paper we focus on interpretable models for binary classification problems [4]. In such a context, ML models can be seen as formalisms to represent Boolean functions that map binary features to one of two classes. Several known classification models can associate predictions with explanations, such as decision lists (DLs), decision sets (DSs) and decision trees (DTs) [5], among others. The size of a ML model plays a critical role in interpretability, with smaller representations leading to more concise and easier to understand explanations. Binary Decision Diagrams (BDDs) are well-known data structures providing compact representations of Boolean functions. They have been used in several domains where Boolean functions play a key role, like logic synthesis and model checking [6]. Given their succinctness, we advocate the use of BDDs as interpretable models for binary classification. Explanations in a BDD-based classifier correspond to paths in a directed acyclic graph (DAG), which are bounded by the number of variables. Unlike generic decision trees, BDDs require a fixed ordering of their variables, which also impacts their size. This allows for more concise explanations to be derived with respect to other models in which an explanation may feature multiple decisions over the same variable. Furthermore, sub-paths sharing should empirically lead to shorter explanations.

[1]A preliminary version was presented in [1]. Part of this work is supported by Partenariato Esteso - RESTART.

A. Related Works

We build on recent works on learning concise machine learning models such as DTs [7] and DLs [8]. The complexity of learning optimal decision trees is properly established in the literature [9]. Decision diagrams are not new in the context of classification [10]. BDDs are used in [11] to classify network packets: the work relies on techniques known in logic synthesis, such as sequential cover and functional decomposition. In this paper we focus on binary classification, using BDDs as an alternative to DTs and other interpretable models that leads to shorter explanations.

B. Contribution

In this paper we propose: (A) A SAT-based approach to learn optimal, i.e., minimum-size, BDD-based classifiers that exhibits perfect accuracy on training data. This technique works well with small problems, but doesn't scale to bigger ones; (B) A heuristic approach for building sub-optimal BDD-based classifiers, focusing on scalability and exploiting BDD compaction techniques to keep the size of the classifier small; (C) An experimental evaluation showing that BDDs can provide a succinct representation, both in term of overall size and in term of size of the explanations.

II. Preliminaries

Boolean functions and formulae A *Boolean function* f over input variables $Vars(f) = \{x_1, \ldots, x_n\}$ is a mapping $f(x_1, \ldots, x_n) : \mathbb{B}^n \rightarrow \{0, 1, -\}$, where $\mathbb{B} = \{0, 1\}$, with 0, 1, and $-$ representing logical *false*, *true* and *don't care*. A *truth assignment* is a mapping from variables in f to truth values. The *onset* is the set of truth assignments v such that $f(v) = 1$, i.e., $onset(f) = \{v \in \mathbb{B}^n | f(v) = 1\}$. The *offset* and the *don't care set* are defined similarly, with $f(v) = 0$, and $f(v) = -$, respectively. The union of onset and offset of f is its *careset*, denoted by $careset(f)$. A Boolean function f

is *completely specified* if its don't care set is empty, otherwise it is *incompletely specified*.

Boolean formulae are symbolic representations of Boolean functions. A *literal* is a, possibly negated, Boolean variable. A *cube* is a conjunction of literals and a *clause* is a disjunction of literals. A conjunction of clauses is a formula in *Conjunctive Normal Form* (CNF). Given a Boolean formula F, the *satisfiability* problem (SAT) is the problem to determine if there exists a truth assignment v so that F evaluates to true under v, *i.e.*, $v \models F$.

The *cover* of f, denoted by C, is a set of cubes that includes the onset of f and does not intersect its offset. For completely specified functions, C coincides with the onset. Otherwise, C contains the onset of f and is contained by the union of the onset and don't care set.

We define the *positive* (*negative*) *cofactor* of function f, with respect to a variable x_i, as $f_{x_i} = f(x_i = 1)$ ($f_{\neg x_i} = f(x_i = 0)$). Given a variable x_i, f can be expressed, according to the *Shannon-Boole expansion theorem*, as: $f(x_1, \ldots, x_n) = x_i \wedge f_{x_i} \vee \neg x_i \wedge f_{\neg x_i}$.

Given two completely specified functions f and g, the *generalized cofactor* of f with respect to g, denoted as $f|g$, is the incompletely specified function f_g that has $onset(f_g) = onset(f) \cap onset(g)$ and $dcset(f_g) = offset(g)$.

A binary encoding enc of an arbitrary finite set A is an injective mapping $enc : A \rightarrow \mathbb{B}^n$, with $n \geq \lceil log_2(|A|) \rceil$. If $n = \lceil log_2(|A|) \rceil$ we have a minimum-length encoding. Any $S \subseteq A$ can be represented by its characteristic function χ_S where $\forall (x_1, \ldots, x_n) \chi_S(x_1, \ldots, x_n) = 1$ iff $enc(s) = (x_1, \ldots, x_n)$ and $s \in S$, given a binary encoding enc.

Binary Decision Diagrams *Reduced-Ordered Binary Decision Diagrams* (ROBDDs or simply BDDs) [12] are a popular way to represent Boolean functions. A BDD is a rooted DAG with two terminal nodes representing values 0 and 1. Internal nodes represent decisions. Each of these nodes is labelled with a variable and has two children. Any edge assigns a truth value to the parent variable. Being *reduced*, isomorphic subgraphs are merged and any node whose two children are isomorphic is deleted. Being *ordered*, the order in which variables appear along all paths from the root is the same. Given a variable ordering, the representation of a Boolean function as a BDD is canonical (unique). Though in principle the size has an exponential worst case [12], many techniques have been developed to find variable orderings that lead to compact BDDs [13].

Classification Problems We follow the notation established by previous works on classification models [14]. Consider a set of *features* $\mathcal{F} = \{x_1, \ldots, x_K\}$. We assume all features to be binary, if necessary standard encoding techniques such as one-hot-encoding and binarization can be used to handle non-binary categorical and numeric features [15]. A literal on a feature x_r, represented as x_r or $\neg x_r$, is used to denote that the feature takes value 1 or 0, respectively. The *feature space* is defined as $\mathbb{F} = \prod_{r=1}^{K} \{x_r, \neg x_r\}$. Let $\mathcal{C}$ be a discrete set of classes, we assume there exists a *target classifier function* $\phi : \mathbb{F} \rightarrow \mathcal{C}$ mapping points in the feature space to classes. Given a training set $\mathcal{E} = \{\epsilon_1, \ldots, \epsilon_M\}$ in which each training instance (or *example*) $\epsilon_i \in \mathcal{E}$ is $2-$tuple $(\mathcal{L}_i, c_i)$, with $\mathcal{L}_i$ a set of literals denoting the point in the feature space associated with the example and $c_i \in \mathcal{C}$ the class (or *prediction*) to which the example belongs, the classification problem is the problem to compute an approximate function $\hat{\phi} : \mathbb{F} \rightarrow \mathcal{C}$ that matches ϕ on training data and generalizes suitably well on unseen test data. In this paper we focus on binary classification, *i.e.*, problems with exactly two classes $\mathcal{C} = \{\oplus, \ominus\}$, with $\oplus$ ($\ominus$) representing 1 (0). The training set $\mathcal{E}$ is partitioned into $\mathcal{E}^+$ and $\mathcal{E}^-$.

In case of multi-class classification, one could apply reduction techniques [16] in order to transform the original problem into a binary classification one.

III. Computing Optimal BDD Classifiers

This section describes a SAT model for computing the smallest BDD consistent with a given dataset. Some ideas are adapted from [7], [17]. We present an iterative approach in which, at each step, we consider the problem of determining whether there exists a BDD of a given size N consistent with training data, encoding it as a SAT problem. We first perform an exponential search, starting from $N = 3$ and doubling it until SAT is found, to identify a suitable range containing the optimum. We then perform binary search on such a range to determine the minimum BDD size.

A. Notation

At each step we consider a BDD of size N. We use indices n, i to refer to BDD nodes. The root node is identified by 1 whereas terminal nodes are identified by $N-1$ and N. We associate the left (right) child node with the decision value 0 (1). We use indices k, j, l to refer to features. We use index m to refer to instances. Table I collects variables and constraints described in the following sections, along with their upper bounds in terms of either variables or clauses. Summation constraints have been encoded according to [18].

TABLE I: SAT model: variables and constraints

1)	a_{kn}	$1 \leq k \leq K, 1 \leq n < N-1$	$\mathcal{O}(NK)$
2)	p_{in}	$1 \leq i < n \leq N, i < N-1$	$\mathcal{O}(N^2)$
3)	lc_{ni}	$1 \leq i < n \leq N, i < N-1$	$\mathcal{O}(N^2)$
4)	rc_{ni}	$1 \leq i < n \leq N, i < N-1$	$\mathcal{O}(N^2)$
5)	pr_{kl}	$1 \leq k \leq K, 1 \leq k \leq K$	$\mathcal{O}(K^2)$
6)	$\sum_{1 \leq k \leq K} a_{kn} = 1$	$1 \leq n < N-1$	$\mathcal{O}(NK)$
7)	$\sum_{1 \leq i < n} p_{in} \geq 1$	$1 < n \leq N$	$\mathcal{O}(N^2)$
8)	$p_{in} \leftrightarrow lc_{ni} \vee rc_{ni}$ $\neg lc_{ni} \vee \neg rc_{ni}$	$1 \leq i < n, 1 < n \leq N$	$\mathcal{O}(N^2)$
9)	$\sum_{i < n \leq N} lc_{ni} = 1$ $\sum_{i < n \leq N} rc_{ni} = 1$	$1 \leq i < N-1$	$\mathcal{O}(N^2)$
10)	$pr_{kl} \vee pr_{lk}$	$1 \leq k, l \leq K, k \neq l$	$\mathcal{O}(K^2)$
11)	$a_{kn} \wedge p_{in} \rightarrow (a_{li} \rightarrow pr_{lk})$	$1 \leq k, l \leq K$ $1 \leq i < n < N-1$	$\mathcal{O}(N^2K^2)$
12)	$pr_{kk} = 0$	$1 \leq k \leq K$	$\mathcal{O}(K)$
13)	$pr_{kl} \wedge pr_{lj} \rightarrow pr_{kj}$	$1 \leq k, l, j \leq K$ $k \neq l, l \neq j, j \neq k$	$\mathcal{O}(K^3)$
14)	e_{mn}	$1 \leq m \leq M, 1 \leq n < N$	$\mathcal{O}(NM)$
15)	$e_{m1} = 1$	$1 \leq m \leq M$	$\mathcal{O}(M)$
16)	$e_{mN-1} = 1$	$\forall \epsilon_m \in \mathcal{E}^+$	$\mathcal{O}(M)$
17)	$e_{mN} = 1$	$\forall \epsilon_m \in \mathcal{E}^-$	$\mathcal{O}(M)$
18)	$e_{mN} \leftrightarrow \neg e_{mN-1}$	$1 \leq m \leq M$	$\mathcal{O}(M)$
19)	$e_{mn} \leftrightarrow \bigvee_{\substack{1 \leq i < n \\ 1 \leq k \leq K}} e_{mi} \wedge a_{ki} \wedge ec(n, i, m, k)$	$1 < n \leq N$ $1 \leq m \leq M$	$\mathcal{O}(N^2KM)$

B. Guessing a valid BDD

We will use a SAT oracle to guess a BDD, with the previously introduced variables 1) through 5): $a_{kn} = 1$ iff feature x_k is associated with BDD node n, $p_{in} = 1$ iff node i is the parent node of node n, $lc_{ni} = 1$ iff node n is the left child of node i, $rc_{ni} = 1$ iff node n is the right child of node i, and $pr_{kl} = 1$ iff feature x_k precedes feature x_l in ranking.

We define the constraints representing a valid BDD, with N nodes, defined on K variables (or features), with the following meaning. 6) Each (non-terminal) node is associated with one feature. 7) Each BDD node, except for the root, has at least one parent. Also, node N must not have $N-1$ as its parent, therefore we add the unit clause $\neg p_{N-1N}$. 8) A node has to be left or right child of its parent(s), and it cannot be both left and right child of another node. 9) Any non-terminal node, has exactly one left child and one right child 10) Relative order of features in ranking is consistent. 11) For every parent-child pair, the feature of the parent node precedes the feature of the child in the ranking. 12) A feature cannot precedes itself in ranking. 13) Precedence in ranking is transitive.

C. Ensuring a BDD consistent with the dataset

Given a consistent dataset, a BDD-based classifier is *perfect* if for each instance it has one path evaluating it to the correct prediction and no other path leading to the incorrect one. Comparing perfect BDD-based classifiers with sparse ones (trading-off accuracy for model size) to assess whether perfect models incur in overfitting is out of this paper scope, but it could be the subject for a future work.

To ensure consistency with the dataset, we introduce new variables e_{mn}, indicating whether a node n *evaluates* an instance m, and new constraints with the following meaning. 15) The root node evaluates all instances. 16) and 17) Terminal node 0 (1) evaluates each instance with prediction value 0 (1). 18) An instance is evaluated at terminal node 1 or 0, not both. 19) Node n evaluates instance ϵ_m iff a parent i of n evaluates ϵ_m and n is either a left or right child of i according to the value of the feature assigned to i in ϵ_m. ec (*evaluating child*) is a function $ec : \mathbb{N}^4 \mapsto Var$ mapping a tuple (n, i, m, k) to variable lc_{ni} (rc_{ni}) if feature k takes value 0 (1) for ϵ_m.

The encoding size is bound by the worst-case constraint among 11), 13) and 19), depending on M, K and N values.

D. Explaining BDDs

Some ML models are widely accepted to be intrinsically interpretable, allowing explanations to be directly extracted from the model. Similarly to DTs, BDDs can considered intrinsically interpretable as well due to their simple structure. Explanations can be extracted from BDDs by simply collecting the decisions made at each non-sink node along the path evaluating an instance. However, intrinsic explanations extracted from interpretable models may be subject to some degree of redundancy [19]. A growing body of research in recent years have been studying how to extract more rigorous and concise explanations from ML models [20]. Formal explanation approaches have been proposed for a variety of models, including DTs. Since BDDs share the same sequential split-based approach to decision-making of DTs, the methods for deriving formal explanations described in [19] can be applied.

IV. Classification by BDD-based cover

The proposed approach tackles the scalability issues of the method described in Section III, by trading off size optimality of the classifiers for faster learning times. Our approach, related to [11], investigates the use of BDD minimization strategies stemming from logic synthesis, applied to BDD-based classifiers. We aim at finding an optimal cover Φ for an incompletely specified function, derived from the training set. We consider the two partitions of the training set as the onset and offset of an incompletely specified classifier function, whose *care set* coincides with the training set itself. The main challenge is to obtain a reasonably high-accuracy classifier while keeping the BDD size small. Classification accuracy highly depends on the quality of the training set, therefore we mainly focus on the second aspect. We exploit generalized cofactor operators: a generalized cofactor [21] is an operator $F(x)|C(x)$, where F is the "cofactored" term, and C is the cofactoring term. The result is a function such that $(F(x)|C(x)) \wedge C(x) = F(x) \wedge C(x)$.

The generalized cofactor combines two effects: generating a new function equivalent to the original one within the onset of the cofactoring term C and compacting the BDD size of $F|C$ by removing unnecessary nodes. It can be seen as a logic optimization operator, as it considers F as an incompletely specified function, where C is the "care set". A freedom of choice for F is thus represented by $\neg C$.

A. The BDD-based classifier

We present two approaches to compute sub-optimal BDD-based classifiers:

- a first one inspired by the "sequential cover" method described in [11], followed by a cofactoring step
- a second one in which we interleave cofactoring steps within the loop encoding each new dataset instance.

As we focus on the binary case, the training set $\mathcal{E} = \{e_1, \ldots, e_M\}$ is split in two partitions $\mathcal{E}^+$ and $\mathcal{E}^-$. Sets are represented by their characteristic functions $\mathcal{E}(x)$, $\mathcal{E}^+(x)$ and $\mathcal{E}^-(x)$. With abuse of notation, we do not formally distinguish a set from its characteristic function. Given an incompletely specified function with onset $\mathcal{E}^+$ and offset $\mathcal{E}^-$, the classifier can be considered as a function obtained from the dataset by providing a completely specified extension of that function. Note that the care set of the incompletely specified function is $Care(x) = \mathcal{E}(x) = \mathcal{E}^+(x) \vee \mathcal{E}^-(x) = \vee_{i\in[1,M]} e_i(x)$.

The target classifier function Φ can be generated from $\mathcal{E}^+$ (or $\mathcal{E}^-$) in multiple ways, with impact on its accuracy. We can produce a trivial extension by applying the generalized cofactor operator: $\Phi^+(x) = \mathcal{E}^+(x)|\mathcal{E}(x)$ and $\Phi^-(x) = \mathcal{E}^-(x)|\mathcal{E}(x)$, with $|$ representing a generalized cofactor operators, that could be either "constrain" or "restrict". In practice, restrict is often better for size optimization, whereas constrain is more regular for symbolic manipulations.

B. Cover approach with cofactor

The operation is an instance of a cover problem in which the target function (Φ^+) can be seen as the cover of an incompletely specified function with onset $\mathcal{E}^+$ and offset $\mathcal{E}^-$. As in [11], we start by generating Φ^+ with the so called sequential approach, which iterates on cubes of $\mathcal{E}^+$, individually optimized. Then we simplify and expand it by taking the generalized cofactor of (the BDD of) $\mathcal{E}^+$ w.r.t. $\mathcal{E}$.

Our experiments show that cofactors may have some edge in terms of BDD size, whereas accuracy seems to be more dependent on the quality of the dataset, rather than on the extra simplification step. Results show a relevant compaction effect, both in BDD size and set of support variables. Due to the two-step process, first computing the full classification function, then taking the cofactor, we can expect a difference in the application of the restrict and constrain cofactors.

C. Interleaved cofactoring steps

We also implemented a variant of the sequential cover that interleaves cofactoring-based simplification steps with training set instances enumeration, computing both the the careset and the classification function incrementally.

Cubes in $\mathcal{E}(x)$ are iteratively picked, by alternating cubes in $\mathcal{E}^+$ and $\mathcal{E}^-$. Each new cube is added to (or excluded from) the classifier function, added to the careset and then the classifier function is cofactored with the newly updated careset. More formally, the target cover is initialized as $\Phi_0^+(x) = 0$, then it is iteratively updated by each element $e_i(x) \in \mathcal{E}(x)$. If $e_i \in \mathcal{E}^+$, then $\Phi_i^+(x) = \Phi_{i-1}^+(x) \vee e_i(x)$ otherwise ($e_i \in \mathcal{E}^-$) $\Phi_i^+(x) = \Phi_{i-1}^+(x) \wedge \neg e_i(x)$. Then it is or-ed with the care set $\mathcal{E}_i(x) = \mathcal{E}_{i-1}(x) \vee e_i(x)$. $\Phi_i^+(x)$ is finally cofactored with the care set $\Phi_i(x) = \Phi_{i-1}(x)|\mathcal{E}_i(x)$.

The advantage of interleaving can be seen in terms of "early" simplification, where newly introduced training set terms are simplified as soon as they are included in the "partial" classifiers. This reduces the difference between constrain and restrict, as they doesn't differ on cofactoring a "cube".

V. Experimental Results

The proposed SAT-based approach has been implemented in Python using the PySAT [22] framework and Glucose3 [23] as a SAT oracle. The heuristic BDD synthesis approach has been implemented in PdTRAV [24].We performed experiments on a set of benchmarks derived from the Penn Machine Learning Benchmarks repository [25].

Focusing on binary classification problems, a selection of datasets has been binarized using standard techniques, such as *one-hot encoding*, to convert all the features to binary. Experiments on learning the optimal BDD-based classifier from the training set were performed on sub-sampled version of the binarized datasets. All other experiments were performed on the complete binarized datasets. To make the search space size more manageable, we use sub-sampling, where a fraction r of samples is randomly selected from each dataset. For each of the selected datasets we sample 20 benchmarks considering various sampling ratio r. Each benchmark is split into a training set and a test with a $80\%/20\%$ ratio. Experiments were run on an 4 cores Intel i7 3370 3.40 GHz workstation with 16 GB of main memory, running Ubuntu $20.04.4$ LTS, using a 3 hours timeout and a 4 GB memory limit.

Table II compares the SAT-based approach with the heuristic one. The leftmost columns report the name, number of features ($\#K$), sampling ratio (r) and the number of samples ($\#M$). Remaining columns report the number of instances solved to optimality ($\#opt$), average optimum BDD size ($\#N_{avg}$), average total and SAT time in seconds ($Time$ and $Time_{SAT}$) and average percentage accuracy ($\%A_{avg}$). For each dataset and sampling ratio r, the results are averaged over 20 runs. Averages are computed on the number of benchmarks for which the optimum was found. As all instance could be solved by the heuristic method, the column $\#opt$ is omitted. While data show that the SAT-based approach offers little scalability, solved instances provide a lower bound for BDD sizes that can be used as a term of comparison for the heuristic approach. Sizes computed by the heuristic approach are in the range $\times 1$ to $\times 5$ the optimal value. With respect to optimal decision trees, as reported in [14], optimal BDD-based classifiers are more compact in terms of average number of nodes. As introduced in section III, the SAT based approach is characterized by a few rather expensive constraints depending on both the number of features and the number of instances of the datasets, thus its practical application may be severely hindered by dataset themselves. At the same time, the heuristic covers tends to be still rather compact, for a fraction of the cost for generating them, with reasonably shallow explanation.

Table III focuses on the heuristic approach and compares the purely "sequential cover" approach (COV), implemented with prime implicant minimization functions, to the one based on cofactoring (COF) and the interleaved variant (INT). Experiments show that cofactoring highly reduces size with respect to COV. Accuracy values changes slightly in relation to the relevance of the chosen dataset. Tables III and II columns share the same meaning. Column $\#N$ describes the number of nodes in the single BDD generated for each run. Columns labelled with max_{len} report the maximum number of features to be tested for classification, *i.e.*, the longest root-to-leaf path in the BDD. Cofactoring based approaches generally win over pure sequential cover, *i.e.*, they lead to shorter explanations in the worst case. The comparison between COF and INT shows generally better results for the former, at the cost of higher computing times. If we take into account both the worst-case path length and the number of features the explanations obtained by either COF or INT uses only between 1% and 16% of the overall number of features, disregarding smaller datasets for which this effect is limited.

VI. Conclusions

In this paper we have tackled the task of learning BDD-based classifiers consistent with a given dataset. We have proposed a SAT-based procedure to learn minimum-sized BDDs, as well as a heuristic approach trading-off size minimality for scalability. We have experimentally observed that the SAT-

TABLE II: Comparison between SAT-based and heuristic approaches.

				SAT					BDD		
$Name$	$\#K$	r	$\#M$	$\#opt$	$\#N_{avg}$	$Time$	$Time_{SAT}$	$\%A_{avg}$	$\#N_{avg}$	$Time$	$\%A_{avg}$
appendicitis	530	0.05	5	20	3.1	743.5	13.1	80%	3.3	3.5	67%
		0.1	10	20	3.5	1692.9	18.3	85%	5.4	6.8	70%
		0.2	21	20	4.4	804.2	50.5	83%	11.2	14.1	71%
breast-cancer	41	0.05	13	20	4.35	1.40	0.1	69%	7.2	1.7	64%
		0.1	26	20	6.5	23.7	16.1	64%	13.9	3.2	61%
		0.2	53	7	8.8	3599.3	3562.1	70%	25.5	4.1	62%
colic	415	0.05	17	20	4.4	1798.6	17.5	64%	8.5	3.8	71%
		0.1	35	17	5.7	2953.7	196.1	76%	17.1	6.5	65%
		0.2	71	1	6.0	2158.8	281.9	66%	34.2	30.1	66%
cleve	395	0.05	15	20	4.1	314.5	7.6	77%	8.5	3.4	73%
		0.1	30	18	8.9	744.7	401.9	70%	15.7	10.9	59%

TABLE III: Results on larger instances solved heuristically.

			COV				COF				INT			
$Name$	$\#K$	$\#M$	$\#N$	$Time$	$\%A$	max_{len}	$\#N$	$Time$	$\%A$	max_{len}	$\#N$	$Time$	$\%A$	max_{len}
australian	1163	552	33664	406.24	68%	158	189	406.24	73%	49	336	66.38	73%	49
breast-cancer	41	202	10844	0.17	65%	25	117	0.17	50%	15	121	0.08	65%	18
cars	711	314	7	9.03	100%	6	7	9.03	100%	6	92	5.91	54%	65
chess	38	3196	17635	1.57	97%	33	469	1.57	90%	24	580	2.10	85%	25
cleve	395	242	10143	18.10	71%	39	100	18.10	66%	29	127	4.22	54%	47
coil2000	654	1023	13284	229.55	84%	51	186	229.55	89%	20	158	54.07	84%	37
colic	415	294	2944	28.35	84%	77	139	28.35	89%	40	158	5.74	84%	37
dis	1105	510	49	59.75	96%	12	9	59.75	96%	7	84	58.59	92%	69
german	1073	511	93005	328.24	70%	93	215	328.24	65%	24	215	55.96	65%	24
hypothyroid	1185	511	28811	387.60	88%	126	150	387.60	81%	120	249	68.53	60%	120
mushroom	112	8204	100	16.30	100%	18	22	16.30	100%	18	128	26.66	100%	18

based approach, though exhibiting poor scalability, is able to compute lower bounds. Our results indicate that the optimal BDD size and the size of BDDs obtained using the heuristic approach are quite close. Furthermore, experimental results, albeit preliminary, seem to indicate that classification accuracy is more related to the quality of the data set rather than to the effort spent on minimizing the size of the classifier.

REFERENCES

[1] G. Cabodi, P. E. Camurati, A. Ignatiev, J. Marques-Silva, M. Palena, and P. Pasini, "Optimizing binary decision diagrams for interpretable machine learning classification," in *DATE*, 2021, pp. 1122–1125.

[2] B. Casey, A. Farhangi, and R. Vogl, "Rethinking explainable machines: The GDPR's "right to explanation" debate and the rise of algorithmic audits in enterprise," *Berkeley Technology Law Journal*, vol. 34, 2018.

[3] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 93:1–93:42, 2019.

[4] C. Molnar, *Interpretable Machine Learning*. Leanpub, 2020. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[5] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, p. 81–106, Mar. 1986.

[6] R. E. Bryant, "Binary decision diagrams," in *Handbook of Model Checking*, E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds. Springer, 2018, pp. 191–217.

[7] N. Narodytska, A. Ignatiev, F. Pereira, and J. Marques-Silva, "Learning optimal decision trees with sat," in *IJCAI*, 7 2018, pp. 1362–1368.

[8] J. Yu, A. Ignatiev, P. L. Bodic, and P. J. Stuckey, "Optimal decision lists using SAT," *CoRR*, vol. abs/2010.09919, 2020.

[9] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is NP-complete," *Inf. Process. Lett.*, vol. 5, no. 1, pp. 15–17, 1976.

[10] A. Oliveira, A. Sangiovanni-Vincentelli, and J. Shavlik, "Using the minimum description length principle to infer reduced ordered decision graphs," in *Machine Learning*, 1996, pp. 23–50.

[11] N. Narodytska, L. Ryzhyk, I. Ganichev, and S. Sevinc, "Bdd-based algorithms for packet classification," in *FMCAD*, 2019, pp. 64–68.

[12] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[13] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on Computers*, vol. C-27, no. 6, pp. 509–516, 1978.

[14] N. Narodytska, A. Ignatiev, F. Pereira, and J. Marques-Silva, "Learning optimal decision trees with SAT," in *IJCAI*, 2018, pp. 1362–1368.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[16] A. Beygelzimer, H. Daumé, J. Langford, and P. Mineiro, "Learning reductions that really work," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 136–147, 2016.

[17] M. Janota and A. Morgado, "Sat-based encodings for optimal decision trees with explicit paths," in *SAT*, 2020, pp. 501–518.

[18] C. Sinz, "Towards an optimal cnf encoding of boolean cardinality constraints," in *Principles and Practice of Constraint Programming - CP 2005*, P. van Beek, Ed. Springer Berlin Heidelberg, 2005.

[19] Y. Izza, A. Ignatiev, and J. Marques-Silva, "On tackling explanation redundancy in decision trees," *Journal of Artificial Intelligence Research*, vol. 75, pp. 261–321, 09 2022.

[20] A. Ignatiev, N. Narodytska, and J. Marques-Silva, "Abduction-based explanations for machine learning models," *CoRR*, vol. abs/1811.10656, 2018.

[21] H. Touati, H. Savoj, B. Lin, R. Brayton, and A. Sangiovanni-Vincentelli, "Implicit state enumeration of finite state machines using bdd's," in *1990 IEEE ICCAD*, 1990, pp. 130–133.

[22] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," in *SAT*, 2018, pp. 428–437.

[23] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern sat solvers," in *IJCAI*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, p. 399–404.

[24] G. Cabodi, S. Nocco, and S. Quer, "Benchmarking a model checker for algorithmic improvements and tuning for performance," *Formal Methods in System Design*, vol. 39, no. 2, pp. 205–227, 2011.

[25] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "Pmlb: a large benchmark suite for machine learning evaluation and comparison," *BioData Mining*, vol. 10, no. 1, p. 36, Dec 2017.