IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 18, 2021, accepted August 1, 2021, date of publication August 3, 2021, date of current version August 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3102239

A Review and Experimental Comparison of Multivariate Decision Trees

LEONARDO CAÑETE-SIFUENTES, RAÚL MONROY, AND MIGUEL ANGEL MEDINA-PÉREZ
Tecnologico de Monterrey, School of Engineering and Science, Atizapán de Zaragoza, Estado de México 52926, Mexico
Corresponding author: Raúl Monroy (raulm@tec.mx)

The work of Leonardo Cañete-Sifuentes was supported in part by the National Council of Science and Technology of Mexico under a CONACYT Scholarship Grant.

ABSTRACT Decision trees are popular as stand-alone classifiers or as base learners in ensemble classifiers. Mostly, this is due to decision trees having the advantage of being easy to explain. To improve the classification performance of decision trees, some authors have used Multivariate Decision Trees (MDTs), which allow combinations of features when splitting a node. While there is growing interest in the area, recent research in MDTs all have in common that they do not provide adequate comparison of related work: they do not consider relevant rival techniques, or they test algorithm performance in an insufficient number of databases. As a result, claims have no statistical sustain and, hence, there is a lack of general understanding of the actual capabilities of existing MDT induction algorithms, crucial to improving the state-of-the-art. In this paper, we report on an exhaustive review of MDTs. In particular, we give an overview of 37 MDT induction algorithms, out of which we have experimentally compared 19 of them in 57 databases. We provide a statistical comparison in all databases and subsets of databases according to the number of classes, number of features, number of instances, and degree of class imbalance. This allows us to identify groups of top-performing algorithms for different types of databases.

INDEX TERMS Supervised classification, decision trees, multivariate decision trees, machine learning.

I. INTRODUCTION

Decision trees (DTs) are popular classifiers, partly because their models are easy to explain and because they show remarkable performance. DTs' popularity has further increased due to the increasing need of using white-box decision models: experts need to understand a model because in several practical problems it is mandatory to explain classification results [1]. Decision tree performance is highly competitive through the use of ensembles; in a recent survey [2], Random Forest [3] and eXtreme Gradient Boosting (XGBoost) [4] are among the top-ranked algorithms. Some applications of DTs or DT-based classifiers in such context include: predicting student dropout in subscription-based online learning environments [5], exploring customer purchasing patterns to evaluate the influence of product photos on sales [6], and evaluating the suitability of behavior change techniques in the context of mobile health applications [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

A decision tree is a graph with a tree structure that has a single root node with directed links (branches) to children nodes that may also have branches to other nodes. The terminal nodes, which do not have any branches, are commonly called leaves [8]. Each branch is tagged with a test, which evaluates to `true` or `false` for each object. For branches coming out of the same node, the tests define a partition of the database; so, for each object, one and only one of the tests evaluate to `true`. The tuple of tests tagging branches from a node is known as a split because they are used to split the objects in a node into disjoint subsets during tree construction. Each subset of objects is assigned to a different child node. We also use split as a verb; to split a node is to select a split and generate the corresponding children nodes.

According to the number of features considered in a split, we can categorize decision trees into Univariate Decision Trees (UDTs) and Multivariate Decision Trees (MDTs). UDTs use only one feature in a univariate split (e.g., $weight > 60$, $weight \leq 60$), while MDTs use more (e.g., $2 * height + 3 * weight > 40$, $2 * height + 3 * weight \leq 40$). For decision tree classification, multiple authors have shown that MDTs achieve better accuracy than UDTs [9], [10]. This result is due to MDTs using multivariate splits which, often separate the classes better than using univariate relations. As a result, publications in MDT induction algorithms have proliferated, with almost 30 algorithms introduced between 1977 and 2019 (See Figure 1).

Currently, there is not any comprehensive comparison to determine the relative performance of existing MDTs, let alone identifying the top ones. This is both because there are no surveys about MDTs, and because recent papers introducing MDTS suffer from one or two main shortcomings, in terms of the comparison of previous work: authors do not compare their algorithm with relevant rival techniques, or they do so but not in enough databases, and hence results are insufficient for statistically validating the underlying hypothesis.

Our goal with this paper is to fill in this gap; that is, we aim to evaluate the relative merit of MDT induction algorithms to identify how they compare one another. We hope that our findings help the community to select an MDT to use in a particular context.

To accomplish our goal, first, we have conducted a thorough review of MDT induction algorithms. Our review includes 37 MDT induction algorithms and is organized using an extension of the taxonomy proposed by Yildiz *et al.* [11], which groups algorithms into analytical and iterative.

Next, we conducted a thorough experimental comparison of prominent MDT induction algorithms surveyed in this paper. Our experimentation involved 19 MDTs, all of which are intended for general-purpose classification. We have tested these algorithms, using the implementations provided by their respective author(s), against 57 databases from the UCI repository [12]. The databases were carefully selected to ensure diversity. The largest databases have up to 20,000 objects and 856 features. While we have conducted a fair comparison of the studied MDT algorithms under study, future analysis of MDTs may involve more technological aspects, such as evaluating how well an MDT scales up in a large database, including ultra-high dimensional data.

We evaluated the algorithms on their classification performance according to the Area Under the Curve (AUC) of a Receiver Operating Characteristics (ROC) curve since it is robust to class imbalance [13]. We made a statistical comparison of the algorithms using the Bayesian signed-rank test (see Section IV-B). We apply the statistical test in all databases and subsets of databases according to their number of classes, number of features, number of objects, and degree of class imbalance. This way, we were able to identify the top-performing algorithms for each group of databases.

Our conclusions are stronger than any other found in the literature since we compare nearly four times the number of algorithms as the most thorough MDT study, which compared 5 algorithms in 20 databases [11]. We also compare the algorithms in four more databases than the reviewed study with the largest number of databases [14].

Our main contributions in this paper are:

- We provide a sound and extensive review of MDT induction algorithms.
- We provide the most extensive MDT comparison, with results that are sustained through statistical tests.
- We identify groups of top-performing MDTs for all databases and subsets of databases with common characteristics; these groups are so that their median probability of winning against other algorithms is high and their median probability of losing is low.

The rest of the document is organized as follows. In Section II, we present the notation used through the document and the taxonomy used to organize the MDTs. In Section III, we review MDT induction algorithms, show the widespread limitations of recent papers when comparing their algorithm to previous MDTs, and motivate the need for a thorough survey in the subject, hence motivating this paper. Next, in Section IV, we present the methodology used in our experimental comparison of MDTs, describe the databases, explain how we selected the MDT induction algorithms for our statistical comparison, and describe the measures and statistical tests used to compare the algorithms. In Section V, we provide a statistical comparison of the 19 selected MDT induction algorithms in 57 databases. Finally, in Section VI, we present our conclusions.

II. PRELIMINARIES

To organize our review, we categorize the algorithms by extending the taxonomy of Yildiz *et al.* [11]. We now need to introduce some notation and describe two distinctive elements of MDT induction: the form of candidate splits and the concept of feature selection. This notation will help us understand the taxonomy presented in Section II-B.

A. NOTATION

A training database $\mathbf{D}$ is assumed here to be composed of n instances with m real-valued features. An arbitrary instance is represented by the vector $x = [x_1, x_2, \ldots, x_m]$, with $x_j \in \Re, \forall j \in F$, where $F = \{1, 2, \ldots, m\}$ is the index set of features. Each x_j represents the value that the feature with index $j \in F$ takes for an arbitrary instance x. Each instance is tagged with a class from a predefined set of K classes $C = \{C_1, C_2, \ldots, C_K\}$.

A candidate split for a Univariate Decision Tree (UDT) takes the form $x_j \leq v, x_j > v$, where v is called a univariate split point. So, we can vary the selected feature (represented by the index $j \in F$) and the split point v to generate candidate splits. In comparison, a Multivariate Decision Tree (MDT) considers multiple features, and there may be multiple coefficients involved in the combination of features.

The most common multivariate splits for MDTs are linear splits. Given a subset of features $F' \subseteq F$, for a binary MDT, a linear split takes the form $\sum w_j x_j \leq v$, $\sum w_j x_j > v$, with $w_j \in \Re, \forall j \in F'$. For each feature with index $j \in F'$, w_j is its corresponding weight coefficient in the linear combination.

The weight coefficients of the linear combination are $w = [w_1, w_2, \ldots, w_{|F'|}]$, and the scalar v is called the split point.

To find candidate splits, MDT induction algorithms need to find values for w, v, and F'. Searching for F' is an optional step called feature selection; most algorithms lack feature selection and use all features in a linear combination, $F' = F$. The approaches used to search w, v, and F' are part of the taxonomy we will present.

B. TAXONOMY

To organize our discussion on MDT algorithms, we follow and extend the taxonomy proposed by Yildiz *et al.* [11]. The taxonomy groups the algorithms according to split type, approach to multi-class problems, the method for finding the weight coefficients w, method to find the split point v, branching factor, and split evaluation function. We extend the taxonomy by adding the feature selection strategy, if any.

- **Split type**. There are three possible split types: univariate, multivariate with linear combinations, and multivariate with non-linear combinations. Some MDT induction algorithms use different split types in different nodes.
- **Approach to multi-class problems**. When making a split, existing MDT induction algorithms have been designed to deal either only with two-class problems or with multi-class problems. From those of the latter, some deal with multi-class problems by transforming them into two-class problems.
- **Feature selection**. MDT induction algorithms either use or do not use feature selection for multivariate splits. The algorithms that use feature selection find multivariate splits using subsets of features. Most feature selection algorithms rely on a greedy search. For example, Sequential Forward Selection (SFS) begins with an empty set of features $F' = \emptyset$; then, it adds a feature one at a time, provided that a split improves the evaluation function when using the feature in conjunction with all features already in F'. Brodley *et al.* [15] describe other prominent feature selection algorithms used in MDTs.
- **Branching factor**. The branching factor is the number of children of a node: it is either equal to 2 or the number of classes K.
- **Search for w**. The search for the weight coefficients w can be either analytical or iterative.
- **Search for v**. The search for the split point v can also be analytical or iterative.
- **Evaluation function**. Some MDT induction algorithms generate more than one candidate split and need to use an evaluation function to choose one. Hernández *et al.* [16] have conducted an experimental comparison of evaluation functions, where they rank evaluation functions used with C4.5 by the classification performance achieved in terms of accuracy and AUC.

III. RELATED WORK

Since we want to identify common strategies for building MDTs, we have grouped split generation algorithms into two broad categories, according to their strategy for finding w. The first category of algorithms, which we review in Section III-A, use analytical solutions for finding w. The second category of algorithms, which we review in Section III-B, use iterative approaches for finding w. In Section III-C, we briefly discuss algorithms related to MDTs. In Section III-D, we present the widespread limitations found in MDT induction literature regarding the comparison of new algorithms against relevant rival MDTs. Finally, in Section III-E, we present the conclusions of our review.

A. ANALYTICAL MULTIVARIATE SPLIT GENERATION

Analytical algorithms use only analytical calculations to find the weight coefficients w. Some of the algorithms in this category also find the split point v through analytical calculations, while others find v through an iterative algorithm after finding w. Most algorithms in this category use Linear Discriminant Analysis (LDA). There are two different approaches to LDA: Fisher's linear discriminant and Discriminant functions. Table 1 displays how we categorize each of the analytical algorithms considered in our investigation (one algorithm per row) in terms of the taxonomy presented in Section II-B.

We notice that the evaluation measure is left blank in some cases because some analytical algorithms only produce a single candidate split, so there is no need to use an evaluation measure. The reasons algorithms in this category may use a split evaluation measure are an iterative search for the split point v, the usage of feature selection, or deciding between a couple of candidate splits generated only through analytical calculations. For analytical methods, we have identified the following common split generation strategies:

- **Discriminant functions (Section III-A1)**. Algorithms in this category use discriminant functions to generate K-ary decision trees.
- **New features through discriminant functions (Section III-A2)**. Algorithms in this category use discriminant functions to build new features as linear combinations of the original ones. In contrast to the previous category, these algorithms use the original and new features to build binary trees.
- **Fisher's linear discriminant (Section III-A3)**. Algorithms in this category use Fisher's linear discriminant.

1) MDTs USING DISCRIMINANT FUNCTIONS. QUEST, CRUISE, AND GUIDE

Loh *et al.* introduced the Quick, Unbiased, Efficient, Statistical Tree (QUEST) [17]; the Classification Rule with Unbiased Interaction Selection and Estimation tree (CRUISE) [18]; and the Generalized, Unbiased, Interaction Detection and Estimation tree (GUIDE) [24]. The three algorithms use linear discriminant functions to make a split; however, CRUISE builds K-ary trees, while QUEST and GUIDE build binary trees.

QUEST and CRUISE apply Principal Component Analysis (PCA), dropping the principal components with small

TABLE 1. Properties of analytical MDT algorithms. The algorithms at the top are used in the experimental comparison in Section V. The column Split refers to the type of split used in the node: Univariate splits (Uni), Linear multivariate splits (Lin), Non-linear multivariate splits (Non), or any combination of the aforementioned split types. The column multi-class refers to the approach taken when working with multi-class databases: some algorithms can work directly with these databases, some algorithms need to transform the problem into one of two classes, and some algorithms cannot deal with multiple classes. The column *w* refers to the method for finding the weight coefficients *w*. The column *v* refers to the method for finding the split point *v*. The column Br refers to the branching factor, which is either equal to 2 or the number of classes *K*.

Algorithm	Split	Multi-class	Feature selection	w	v	Evaluation function	Br
QUEST [17]	Uni/Lin	Transforms the problem	-	Discriminant functions	Discriminant Functions	-	2
CRUISE [18]	Uni/Lin	Works directly	-	Discriminant functions	Discriminant Functions	-	K
LDT [11]	Lin	Transforms the problem	-	Fisher's discriminant	Fisher's discriminant	-	2
Cline [19]	Lin	Only 2 class	-	Analytical	Analytical	-	2
Zhang's MPSVM [2]	Uni/Lin	Transforms the problem	-	MPSVM	MPSVM	Gini index	2
MHLDT [20]	Uni/Lin	Works directly	SFS	Fisher's discriminant	Exhaustive	Hellinger distance	2
Friedman [21]	Uni/Lin	Only two class	-	Fisher's discriminant	Fisher's discriminant	-	2
Ltree/LgTree [22]	Uni/ Lin	Works directly	-	Discriminant functions	Exhaustive	Info Gain	2
Qtree [22]	Uni / Non	Works directly	-	Discriminant functions	Exhaustive	Info Gain	2
LDTS [23]	Lin	Transforms the problem	Tabu search	Fisher's discriminant	Fisher's discriminant	Entropy	2
SURPASS [10]	Uni/Lin	Transforms the problem	-	Fisher's discriminant	Fisher's discriminant	Entropy	2
GUIDE [24]	Uni/Lin	Works directly	Exhaustive search with 2 features	Discriminant Functions	Discriminant Functions	-	2
Geometric [25]	Lin	Transforms the problem	-	MPSVM	MPSVM	Gini index	2
FDT [26]	Lin	Only 2 class	-	Fisher's discriminant	Exhaustive	Impurity	2
HHCART [27]	Lin	Works directly	-	Analytical	Exhaustive	Impurity	2
Efficient [28]	Lin	Works directly	-	Analytical	Analytical	Impurity	2

eigenvalues; in this way, the algorithms avoid the problem of near singular covariance matrices. The remaining principal components are used to find the splits through linear discriminant functions. A newer version of CRUISE [29] can also fit linear discriminant models in each terminal node.

GUIDE is an improvement upon QUEST and CRUISE. The main difference regarding multivariate splits is that GUIDE only allows for linear multivariate splits with two features. GUIDE can also fit models on the leaves; however, it uses kernel and nearest-neighbor node models.

In this paper, we focus on multivariate split generation strategies. However, a more extensive discussion on this family of algorithms can be found in Loh's survey [30].

2) MDTs GENERATING FEATURES THROUGH DISCRIMINANT FUNCTIONS. LTREE, QTREE, AND LgTree

Ltree [22], Qtree, and LgTree [31], in each node, generate discriminant functions for the classes with a number of objects exceeding two times the number of features. The difference between the algorithms is that Ltree uses linear discriminants, Qtree uses quadratic discriminants, and LgTree uses logistic discriminants (which results in linear splits). At each node, the discriminant functions are used to construct new features by projecting the data onto them. An exhaustive search is made to find a split for each feature, including the original features, features constructed in previous nodes, and features constructed in the current node.

3) FISHER's LINEAR DISCRIMINANT. FRIEDMAN, LDTS, SURPASS, LDT, FDT, AND MHLDT

Friedman [21], Linear Discriminant and Tabu Search (LDTS) [23], Scaling Up Recursive Partitioning with Sufficient Statistics (SURPASS) [10], Linear Discriminant Tree (LDT) [11], Fisher's Decision Tree (FDT) [26], and Multi-class Hellinger Linear Discriminant decision tree (MHLDT), [20] use Fisher's linear discriminant to generate splits. However, MHLDT uses a multi-class version of Fisher's linear discriminant that produces $K-1$ eigenvectors used as candidates for w. MHLDT thus avoids grouping multiple classes into two groups.

LDTS and SURPASS find the split point v through analytical methods, while the rest of the algorithms in this section use exhaustive search to find it. The main difference between LDTS and SURPASS is that SURPASS is designed to work with databases large enough to exceed memory size; to work in this context, SURPASS removed the feature selection algorithm used by LDTS.

4) MPSVM. GEOMETRIC DECISION TREE AND ZHANG's MPSVM

Geometric DT [25] generates candidate splits through an analytical algorithm, which the authors claim captures the geometric structure of the data better than algorithms relying on impurity measures. Given two classes, the algorithm uses the Multisurface Proximal SVM (MPSVM) algorithm to find a clustering hyperplane for each class, which is a hyperplane where the average Euclidean distance of all the points in the class to the hyperplane is minimized. If the clustering hyperplanes of both classes are parallel, the authors use the hyperplane between them to split the data. Otherwise, the authors use the angle bisectors of the two hyperplanes as candidate splits and keep the one that minimizes an impurity measure.

Zhang's MPSVM trees [32] borrow the main ideas from Geometric DT; however, they apply regularization methods to deal with singular covariance matrices. There are two versions of Zhang's MPSVM: using Tikhonov regularization or using a univariate split when a singular covariance matrix is found.

5) EFFICIENT DECISION TREE

The authors of Efficient trees [28] propose two analytical algorithms for finding w: selecting w randomly or as the dominant eigenvector of the covariance matrix. For both methods of finding w, the authors project the data onto w, then select v as the median of the projected data.

TABLE 2. Properties of iterative MDT algorithms. The algorithms at the top are used in the experimental comparison in Section V. The column Split refers to the type of split used in the node: Univariate splits (Uni), Linear multivariate splits (Lin), Non-linear multivariate splits (Non), or any combination of the aforementioned split types. The column multi-class refers to the approach taken when working with multi-class databases: some algorithms can work directly with these databases, some algorithms need to transform the problem into one of two classes, and some algorithms cannot deal with multiple classes. The column w refers to the method for finding the weight coefficients w. The column v refers to the method for finding the split point v. The column Br refers to the branching factor, which is either equal to 2 or the number of classes K.

Algorithm	Split	Multi-class	Feature selection	w	v	Evaluation function	Br
CART-LC [34]	Lin	Works directly	Sequential Backward Elimination	Backfitting	Exhaustive	Impurity	2
OC1 [9]	Uni/Lin	Works directly	-	Hill climbing	Exhaustive	Info Gain	2
Omnivariate [35]	Uni/Lin/Non	Transforms the problem	-	Perceptron	Perceptron	Impurity	2
OCT [14]	Uni/Lin	Works directly	Mixed-integer optimization (MIO)	MIO	MIO	Misclassification	2
LMDT [15]	Lin	Works directly	Sequential search	Thermal	Thermal	Misclassification	K
CTNNFE [36]	Non	Transforms the problem	-	Perceptron	Perceptron	MSE	2
SADT [37]	Lin	Works directly	-	Simulated annealing	Simulated annealing	Sum-minority	2
BMDT [38]	Lin	Works directly	-	Neural network	Exhaustive	Impurity	2
APDT [39]	Lin	Works directly	-	Alopex	Alopex	Degree of linear separability	2
Dipolar [40]	Lin	Works directly	Heuristic Sequential Search	Iterative	Iterative	Dipolar criteria	2
FAT/MOC1 [41]	Uni/Lin	Works directly	-	Hill climbing	Info Gain + Maximal margin	Impurity	2
VDT/CDT [42]	Lin	Only 2 class	-	MIO	MIO	Misclassification	2
oRF [43]	Lin	Only 2 class	-	Ridge regression	Ridge regression	Gini index	2
HBDT [44]	Lin	Works directly	-	Iterative	Iterative	Impurity	2
SBT/PT [45]	Lin	Transforms the problem	Maximum bound on w	Budget-aware Classifier	Budget-aware Classifier	-	2 or K
OmniGA [46]	Uni/Lin/Non	Works directly	-	Genetic algorithm	Genetic algorithm	Misclassification	2
DTSVM [47]	Lin	Transforms the problem	-	νSVM	νSVM	-	2
BDTKS [48]	Lin	Works directly	-	K-means	Analytical	-	2

6) CLINE

The authors of Cline [19] propose six analytical algorithms for building MDTs based on finding two points, one for each class, and using the line passing through them as w. The cut point v is selected as the midpoint between the two selected points, projected onto w. The six variants choose the points A and B as follows: CL2 selects the nearest two points from different classes; CL4 uses four points instead of two; CLM selects the mean points of the classes; CLLDA first obtains w through Fisher's vector discriminant, then it selects the nearest two points from different classes after projecting them onto w; CLLVQ finds two centroids using Linear Vector Quantization; and CLMIX tests CLM, CLLDA, and CLLVQ in each node and uses the best one according to the split evaluation function.

7) HHCART

HHCART [27] calculates the eigenvectors of the covariance matrix of each class and uses the eigenvectors to build an $m \times m$ Householder matrix to project the data. Each column of the matrix is a candidate for w. For each candidate w, an exhaustive search is made to find a split point v, keeping the split that minimizes the impurity measure. Originally, HHCART had two variants: HHCART(A), which uses all eigenvectors as candidates for w; and HHCART(D), which uses only the dominant eigenvector as w.

A third version of the algorithm, HHCART(G) [33], is a variation of HHCART(D). HHCART(G) uses the angle bisector from the Geometric DT approach from Section III-A4, instead of the dominant eigenvector as w. For small sample sizes, where the angle bisector cannot be found using the original approach [25], the authors introduced a modified angle bisector. Of the three variations of HHCART, HHCART(G) has the highest average accuracy, and it is the most efficient.

B. ITERATIVE APPROACHES FOR MULTIVARIATE SPLIT GENERATION

In this section, we review iterative algorithms, which given an initial solution for the weight coefficients w, use an iterative procedure to modify them to improve the split evaluation measure. Table 2 displays how each of the iterative algorithms considered in our investigation (one algorithm per row) is categorized in terms of the taxonomy presented in Section II-B. We have identified five general approaches used to search candidate splits through iterative algorithms:

- **Hill climbing (Section III-B1)**. These algorithms use hill climbing or some variation thereof, such as simulated annealing, to search for candidate splits.
- **Linear discriminant functions (Section III-B2)**. These algorithms generate K-ary trees through linear discriminant functions. The linear combination for each discriminant function is obtained through an iterative algorithm.
- **Neural networks (Section III-B3)**. These algorithms use neural networks to generate candidate splits; some of these algorithms generate non-linear splits.
- **Evolutionary algorithms (Section III-B4)**. These algorithms run an evolutionary algorithm to improve an initial population of solutions.
- **Linear programming (Section III-B5)**. These algorithms pose the problem of finding a candidate split as a linear programming problem.

1) HILL CLIMBING AND ANNEALING ALGORITHMS. CART-LC, SADT, OC1, APDT, FAT/MOC1

The oldest iterative algorithm we review, Classification and Regression Trees with Linear Combination (CART-LC), was proposed by Breiman *et al.* [34] as an extension of CART to linear combinations. CART-LC is deterministic. It starts with the best univariate split, normalizes the weight coefficients, and generates two candidate splits by modifying a single weight coefficient at a time by 0.25 and −0.25.

One problem of CART-LC is that it can be easily stuck in local optima because it is deterministic. To solve this problem, Heath *et al.* [37] proposed Simulated Annealing of Decision Trees (SADT), which uses simulated annealing to find the weight coefficients w, so it is more difficult for it to get stuck in local optima.

Murthy *et al.* [9] noted that SADT is computationally expensive, so they propose Oblique Classifier 1 (OC1), an extension to CART-LC that includes two randomization procedures. As CART-LC, OC1 starts with the best axis-parallel split, but it modifies the weight coefficients by small random amounts. The advantage of OC1 over SADT is that it is more efficient.

FAT and Margin OC1 (MOC1) [41] are DTs based on OC1; the main difference from OC1 is that they try to maximize the margin in each node. FAT first builds an OC1 tree; then, in each inner node, the objects are relabeled as *right* or *left* according to the child they fall in. The relabeled objects are linearly separable, so SVM is used to find the hyperplane with maximal margin. MOC1 modifies the split evaluation measure to include the size of the margin.

The Alopex Perceptron Decision Tree (APDT) [39] uses the Alopex algorithm, a variant of simulated annealing to find a split. One difference with the SADT algorithm that uses simulated annealing, is that in APDT, all weights are randomly modified at each step.

2) LINEAR DISCRIMINANT FUNCTIONS. LMDT, oRF

Brodley *et al.* [15] proposed to use linear discriminant functions to generate candidate splits in Linear Machine Decision Trees (LMDT). Usually, discriminant functions use an analytical approach; however, Brodley *et al.* [15] proposed three algorithms for finding the coefficients w_i, v_i iteratively. The first algorithm (RLS) uses the recursive least-squares procedure to find the parameters w_i, v_i; since this approach only works for two classes, the trees generated are binary. The other two algorithms generate k-ary trees, where they consider treating each linear discriminant function as a perceptron. Since convergence is a problem for the perceptron if the objects are not linearly separable, the authors use the Pocket algorithm as a possible solution for not linearly separable problems. As a second solution, the authors use Thermal Training, which is a variation on simulated annealing. The authors' experiments were restricted to two-class databases, where the RLS algorithm outperforms the others in accuracy.

Menze *et al.* [43] proposed Oblique Random Forests (oRF), which builds MDTs using ridge regression, where the regularization parameter λ is adjusted iteratively. The authors note that with $\lambda = 0$ the split is similar to one obtained through discriminant analysis approaches, while $\lambda \gg 1$ results in a split similar to one obtained through principal component analysis.

Both LMDT with recursive least squares and oRF are limited to two-class problems. The other versions of LMDT can work directly with multi-class problems.

3) NEURAL NETWORKS. BMDT, CTNNFE, OMNIVARIATE

Liu *et al.* [38] proposed BMDT, which transforms the problem of inducing binary multivariate decision trees to one of inducing binary univariate decision trees. The algorithm trains a 2-layer feed-forward neural network, where hidden units are used as new features $x_i' = \delta(\sum_{j=1}^{m} w_j^i x_j + w_0^i)$, where w_j^i are the weights from the feature with index $j \in F$ to the hidden unit i, and δ is a non-linear squashing function. A univariate tree is built using the new features, where splits take the form $x_i' \leq v'$. A linear split is obtained by taking the inverse of the squashing function, $\sum_{j=1}^{m} w_j^i x_j + w_0^i \leq \delta^{-1}(v')$; by defining the split point $v = \delta^{-1}(v') - w_0^i$, we can rewrite this split like the linear splits presented in Section II-A ($\sum_{j=1}^{m} w_j x_j \leq v$).

Guo *et al.* [36] proposed Classification Trees with Neural Network Feature Extraction (CTNNFE), which at each node builds a multilayer perceptron to generate non-linear splits. As an extension, Yildiz *et al.* [35] proposed Omnivariate decision trees, which can make univariate splits, linear multivariate splits, and non-linear multivariate splits. To make linear multivariate splits, the Omnivariate algorithm uses a single layer perceptron.

4) EVOLUTIONARY ALGORITHMS. HBDT, OmniGA

Struharik *et al.* [44] proposed the HereBoy DT (HBDT) algorithm, which runs an evolutionary algorithm to find the optimal split point at each node. OmniGA [46] uses a genetic algorithm to generate trees based on Omnivariate trees [35]. The genetic algorithm is used to select the type of split made at each node, optimize the parameters of the split, and prune nodes.

5) LINEAR PROGRAMMING. DIPOLAR, OCT, SBT/PT, VDT/CDT

The authors of Dipolar [40] propose to use the basis exchange algorithm, which is similar to linear programming, to minimize the dipolar criterion function. A dipole is a pair of objects from the database; a pure dipole is one with objects of the same class, while a mixed dipole has objects of different classes. The authors aim to select a hyperplane that divides a high number of mixed dipoles and a low number of pure dipoles, so they define the dipolar criterion function as a weighted sum of the cost of separating pure dipoles and not separating mixed dipoles.

The authors of Optimal Classification Trees (OCT) [14] formulate the problem of building UDTs and MDTs as a mixed-integer optimization problem. The objective function considers a trade-off between accuracy and model complexity.

The authors of Vertical Decision Trees (VDT) and Cutting Decision Trees (CDT) [42] also formulate the problem of building MDTs as a mixed-integer optimization problem. However, VDT is not allowed to grow in width by making each inner node have at least one leaf child. Given that the variables in the optimization problem grow at an exponential rate with respect to the depth of the tree, the authors propose the CDT, in which the number of variables grows linearly with the depth of the tree. In the CDT, only the leaf nodes at maximum depth may be impure.

The authors of Supervised Budgeted Tree (SBT) and Powerset Tree (PT) [45] analyze the error bound of an MDT and conclude that to decrease testing error, they must decrease training error, put a constraint on the weight coefficients, and enlarge the margin in each node. The authors create a Budget-aware classifier, which has these constraints incorporated into an optimization problem. In each node, the Budget-aware classifier is used to generate splits. For two classes, the SBT tree is built in a top-down manner; however, for multiple classes, the PT bottom-up algorithm is used, which generates only one leaf per class.

6) DTSVM

The authors of Decision Tree SVM (DTSVM) [47] build a tree using νSVM to generate splits. Since νSVM is designed to deal with two classes, the authors mention they use a one-versus-others strategy to transform multiple classes into a binary class; however, it is not clear if this strategy is used to generate several trees, or if this strategy is used at each node to generate candidate splits.

After building the MDT, DTSVM encodes each object in a $v = \{0, 1\}^m$ vector, where each element corresponds to an inner node of the tree. If an object passes through node i, then its corresponding value is $v_i = 1$, otherwise $v_i = 0$. This new feature space is used to classify the objects using a linear SVM.

7) BDTKS

Binary Decision Tree based on K-means Splitting (BDTKS) [48] applies K-means with $k = 2$ at each node and uses the centroids to calculate w, v. To calculate w, BDTKS obtains the hyperplane passing through both centroids; then, the centroids are projected onto w, and the midpoint is used as v. The split evaluation function is a modified impurity function that takes into account class imbalance.

C. OTHER DTs WITH MULTIVARIATE DECISIONS

Other types of decision trees use multivariate decisions, such as Soft decision trees, Model trees, and Functional trees. We will briefly describe these trees; however, in this paper, our focus is on MDTs. Comparing these other trees deserves future study.

Soft decision trees [49] and Fuzzy decision trees [50], unlike the DTs we have discussed so far, do not tag branches with binary tests which tell us which branch to follow. Instead, each node has a gating function that gives probabilities or membership degrees to the children nodes. All paths from the root to the leaves are traversed with probabilities assigned by the gating function.

Model trees are univariate trees that make multivariate decisions at each leaf [51], while functional trees are a generalization of MDTs and Model trees [52]. Functional trees allow combinations of features at inner nodes and leaf nodes. Gama [52] compared his algorithm for Functional Trees with one algorithm of the other groups: CRUISE [18] for MDTs, and M5' [53] for Model trees. The algorithms were compared in 30 databases, and a Wilcoxon test with Bonferroni correction showed no significant difference between Functional Trees and the other algorithms. However, the Functional tree was ranked first, then the MDT, and at last the Model tree.

D. LIMITATIONS OF RELATED WORK COMPARISON

When presenting a new classification algorithm, authors should attempt to determine how it compares, in terms of performance, against others that are a reference to the community. Moreover, for this experimental comparison to be statistically sound, all algorithms should be tested in enough databases, observing the conditions of the corresponding hypothesis test. This way, algorithms can be ranked in terms of performance, and so any upcoming algorithm could be compared against only a subset of top-rank algorithms. In the literature about MDT induction, however, we have noticed two widespread limitations on recent papers: the proposed new MDT algorithm is not compared with relevant existing rival MDTs, or the comparison does not involve testing the algorithms in enough databases to support any claim statistically.

Fig. 1 displays each algorithm, ordered in terms of year of appearance, from 1977 to 2018. It aims to convey both the number of times the algorithm in question has been used as a reference to compare the behavior of others (orange bar) and the number of algorithms it was compared against upon introduction (blue bar). We can see that in most papers, authors compare their algorithm against at most a couple of other MDT induction algorithms; the most comprehensive study compares their algorithm against five others. Furthermore, only CART-LC, OC1, QUEST, and LMDT are used as a reference for comparison more than twice. Note that most algorithms have never been used in any experimental comparison.

Fig. 2 is similar to Fig. 1, except that it displays the number of databases used when testing the proposed algorithm. Looking at it, we notice that only 9 algorithms use at least 20 databases. Given that including more databases increases the power of statistical tests [54], and there are publicly available database repositories such as the UCI repository [12], new publications should strive to include more databases in their experimental comparisons.

E. MDT INDUCTION CONCLUSIONS

In this section, we presented a taxonomy that enables us to identify common split generation strategies. We grouped the algorithms into analytical and iterative, according to their approach for generating the weight coefficients w for candidate splits. In both groups, we identified subgroups of strategies for finding w.

Although plenty of algorithms can use a mix of univariate and linear multivariate splits when building a tree, multivariate splits often involve all features because most algorithms lack feature selection. Feature selection is important to

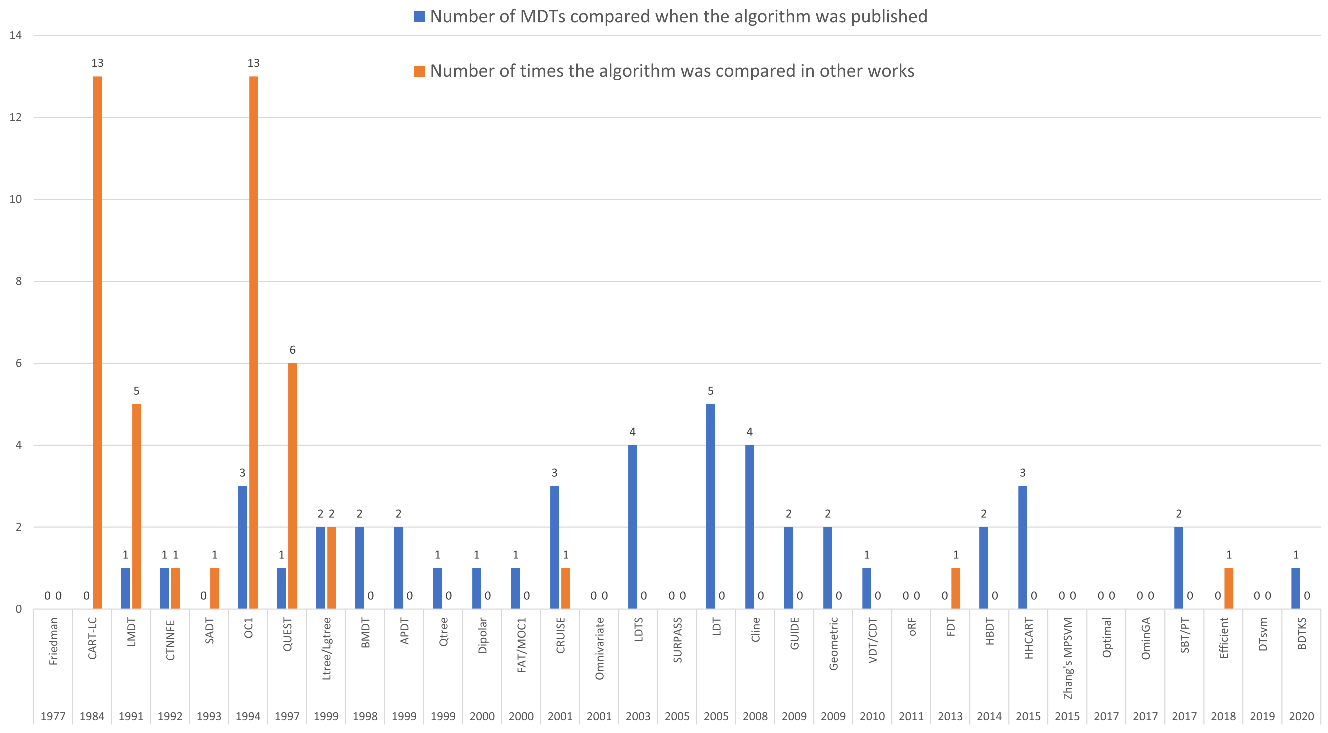

FIGURE 1. MDT induction algorithms sorted by year of publication. For each algorithm, the blue bar represents the number of rival algorithms the authors used in their comparison study. The orange bar represents the number of times the algorithm is compared in papers by other authors.

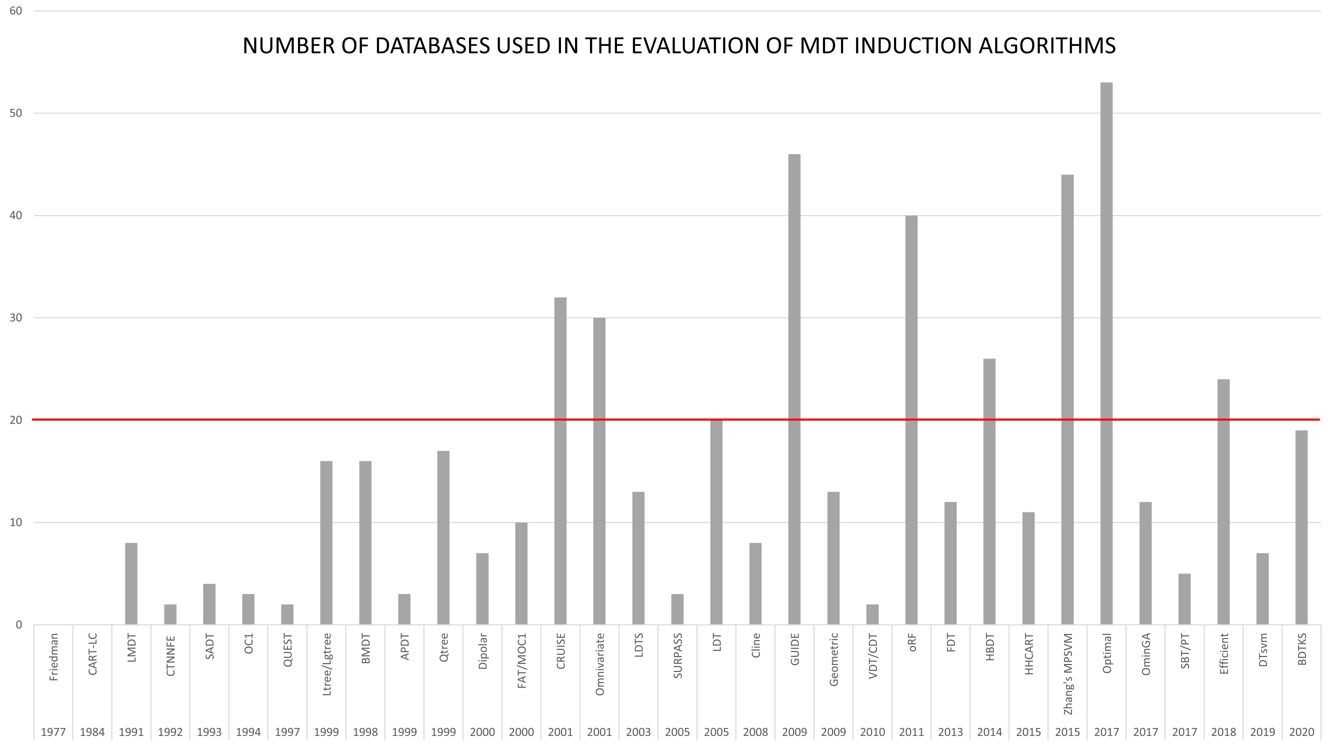

FIGURE 2. Number of databases used in the experimental evaluation of MDT induction algorithms. Only 9 algorithms use at least 20 databases.

keep the models simple and is helpful because class separability is sometimes found on subsets of features [20]. Another problem of some algorithms is that they cannot work with multiple classes, or they group multiple classes into two groups, potentially losing information about class separability found on subsets of features.

TABLE 3. Example confusion matrix obtained by the J48 classifier [55] for the iris database.

	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	49	1	0
Iris-versicolor	0	47	3
Iris-virginica	0	2	48

The widespread limitations on recent papers regarding the comparison of their algorithm against relevant rival MDTs prevent us from readily identifying the best strategies for building MDTs. Therefore, to achieve our goal of evaluating the relative merit of existing MDT induction algorithms, we will use 20 MDT induction algorithms and make a statistical comparison using 57 databases. In the following section, we describe the selection process and databases.

IV. EXPERIMENTAL SETUP

In this section, we present our proposed methodology to accomplish our research objectives of comparing MDTs to each other and identifying the top-performing algorithms. In Section IV-A, we define the measures used to evaluate a classifier. In Section IV-B, we show the methods used to compare classifiers using the measures defined in Section IV-A. In Section IV-C, we describe the databases and algorithms used in this study. Finally, we describe the evaluation protocol in Section IV-D.

A. EVALUATION MEASURES

The measures we use to evaluate classification performance can be obtained from the confusion matrix, which is a result of the classifier applied to a testing database. The confusion matrix is a $k \times k$ matrix, where k is the number of classes. The rows correspond to actual classes and the columns to predicted classes. We show an example of a confusion matrix in Table 3. From the row Iris-setosa, we can see that 49 objects were correctly classified as Iris-setosa, and one object was incorrectly classified as Iris-versicolor. Similar remarks hold for the two other classes.

Let $\mathbf{C}$ be a confusion matrix of size $k \times k$. Each cell c_{ij} in $\mathbf{C}$ counts the number of objects of class i classified as class j.

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1k} \\ c_{21} & c_{22} & \dots & c_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{k1} & c_{k2} & \dots & c_{kk} \end{bmatrix}$$

Accuracy is a popular measure to evaluate a classifier, defined as the number of correctly classified objects divided by the total number of objects in the testing database. It can be obtained by adding the objects of the main diagonal of the confusion matrix over the total number of objects, say n:

$$\mathtt{acc}(\mathbf{C}) = \frac{\sum_{i=1}^{k} c_{ii}}{n} \tag{1}$$

One important drawback of using accuracy is that it does not take into account class imbalance. A database is highly imbalanced if it has many objects of one class compared to the rest of the classes; for such a database, always classifying objects as the class with most objects, so-called the majority class, will result in high accuracy.

Since many real-world databases are imbalanced, we use the Area Under the ROC curve (AUC), which is more insensitive to imbalanced databases [13]. The AUC measure for discrete classifiers for two classes is defined using recall and specificity [13]. Let (C_i, C_j), with $i \neq j$, be any pair of classes, where C_i denotes the Positive class and C_j the Negative one. The number of objects of the Positive class C_i correctly classified, c_{ii}, is the number of True Positives. The number of objects of the Positive class C_i incorrectly classified, c_{ij}, is the number of False Negatives. The number of objects of the Negative class C_j correctly classified, c_{jj}, is the number of True Negatives. The number of objects of the Negative class C_j incorrectly classified, c_{ji}, is the number of False Positives.

The Recall is then defined as the proportion of objects of the positive class correctly classified:

$$\mathtt{r}_{ij}(\mathbf{C}) = \frac{c_{ii}}{c_{ii} + c_{ij}} \tag{2}$$

Specificity is defined as the proportion of objects of the negative class correctly classified:

$$\mathtt{sp}_{ij}(\mathbf{C}) = \frac{c_{jj}}{c_{ji} + c_{jj}} \tag{3}$$

Finally, the AUC for two classes i, j is defined as:

$$\mathtt{auc}_{ij}(\mathbf{C}) = \frac{\mathtt{r}_{ij}(\mathbf{C}) + \mathtt{sp}_{ij}(\mathbf{C})}{2} \tag{4}$$

To extend the definition of AUC to multi-class problems, we take the recommended one versus the others approach [13]. This approach consists of averaging the AUC of all possible pairs of classes as follows:

$$\mathtt{auc}(\mathbf{C}) = \frac{1}{\binom{k}{2}} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathtt{auc}_{ij}(\mathbf{C}) \tag{5}$$

Since many of the databases tested are imbalanced, our performance indicator in this study is AUC. Now, we will describe how to compare algorithms using AUC as an evaluation measure.

B. STATISTICAL COMPARISON

For comparing algorithms in multiple databases, we used the Bayesian signed-rank test as described in the tutorial by Benavoli *et al.* [56]. This test is the Bayesian counterpart to Wilcoxon's test. To understand this test, we briefly describe how two classifiers are compared in a single database.

The Bayesian tests described by Benavoli *et al.* [56] are based on three hypotheses: that classifier A is practically better than B, that the classifiers are practically equivalent,

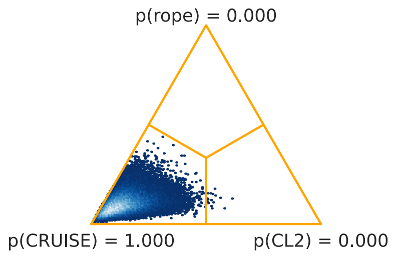

FIGURE 3. Example of Bayesian signed-rank test. Comparison between CRUISE and CL2 in a subset of the 57 databases with a high number of features.

and that classifier B is practically better than A. To calculate the probabilities of the hypotheses for a specific database, the Bayesian correlated t-test is used to obtain a distribution of mean differences of AUC.

The probabilities $\theta_l, \theta_e, \theta_r$ correspond to the integral of the distribution on different intervals: the region $(-\infty, -r)$, where classifier A is practically better than B; the region (r, ∞), where classifier B is practically better than A; and the region $[-r, r]$, where the classifiers are practically equivalent. The interval $[-r, r]$ is known as the region of practical equivalence (rope). Benavoli *et al.* [56] use $r = 0.01$ for accuracy; we will use the same value for AUC given the similarity of the measure for balanced databases.

To compare the classifiers on multiple databases, the Bayesian signed-rank test is used. For this test, a distribution on the probabilities $\theta_l, \theta_e, \theta_r$ is computed by Monte Carlo sampling. For a given sample, there is a bias towards θ_i if $\theta_i > \max(\theta_j, \theta_k)$. So, if for all our samples we have $\theta_i > \max(\theta_j, \theta_k)$, we conclude with a probability equal to 1 that hypothesis i is true. Let us say we conclude that classifier B is practically better than classifier A with a probability equal to 1. This does not necessarily mean that the difference of AUC between classifier B and A is always greater than 0.01. This means that the probability θ_r is always greater than both θ_l and θ_e; in other words, there is always a bias towards classifier B winning.

Benavoli *et al.* [56] visualize $\theta_l, \theta_e, \theta_r$ for each sample using a simplex with vertices $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$. In Figure 3, we show one for a comparison between the classifiers CRUISE and CL2 on a subset of databases with a high number of features. If a point falls in a vertex, then it has $\theta_i = 1$ for the corresponding hypothesis i; in the figure, a point in the left corner is a sample where the difference in AUC between CRUISE and CL2 is greater than 0.01 with a probability equal to 1. There are three regions limited by $\theta_i > \max(\theta_j, \theta_k)$; so the left region corresponds to the case where there is a bias towards CRUISE. In each corner, the proportion of samples falling in the corresponding region is shown; since almost all samples fall in the region where CRUISE is better, we have $p(CRUISE) \approx 1$. Some samples fall in the region where CL2 is better; however, the proportion of those samples is smaller than 1×10^{-3}.

TABLE 4. Distribution of the databases under consideration according to their number of features. The second column shows the distribution for 57 databases, and the third column the distribution for 40 databases. We can test all the 19 MDT induction algorithms only on 40 databases and a reduced set of 15 MDT induction algorithms in 57 databases.

No. of features	No. of databases (57)	No. of databases (40)
< 10	19	17
$10 - 100$	33	20
> 100	5	3

TABLE 5. Distribution of the databases under consideration according to their number of objects. The second column shows the distribution for 57 databases, and the third column the distribution for 40 databases. We can test all the 19 MDT induction algorithms only on 40 databases and a reduced set of 15 MDT induction algorithms in 57 databases.

No. of objects	No. of databases (57)	No. of databases (40)
< 100	3	0
$100 - 1,000$	29	27
$> 1,000$	25	13

TABLE 6. Distribution of the databases under consideration according to their number of classes. The second column shows the distribution for 57 databases, and the third column the distribution for 40 databases. We can test all the 19 MDT induction algorithms only on 40 databases and a reduced set of 15 MDT induction algorithms in 57 databases.

No. of classes	No. of databases (57)	No. of databases (40)
2	25	18
> 2	32	22

C. DATABASES AND EVALUATED ALGORITHMS

We have found 57 numerical databases without missing values from the UCI repository [12]. The databases are diverse, with varying numbers of objects, number of features, number of classes, and degree of imbalance. The full description of the databases can be found in Appendix A, where we can verify the diversity of the databases. However, we summarize key characteristics of the databases in Tables 4, 5, and 6.

In the following section, we compare 19 implementations of MDT induction algorithms using the databases and algorithm comparison methods described in this section. The classifiers compared include seminal MDTs that were designed for general-purpose classification: CART-LC and OC1. The rest of the classifiers are also for general-purpose classification, and the original authors tested them in diverse databases, such as the ones of the UCI repository.

We used the original author's implementation for each classifier included. The implementations were publicly available online, or the authors were kind enough to share with us an implementation for academic purposes. The authors of SBT/PT [45] shared with us the implementation for their classifier; however, the implementation works only for two-class databases and the published results are also only for two-class databases.

We compare the following analytical algorithms, which are at the top of Table 1: QUEST [17], CRUISE [18], LDT [11], six variants of CLINE [19], five variants of MPSVM [2], and MHLDT [20]. We also compare the following iterative algorithms, which are at the top of Table 2: CART-LC [34], OC1 [9], Omnivariate [35], and Optimal [14].

D. EVALUATION PROTOCOL

For each algorithm $a \in \mathbf{A}$ listed in Section IV-C, we executed a in each dataset $d \in \mathbf{D}$ using 5-fold Distribution Optimally Balanced-SCV (DOB-SCV). The k-fold DOB-SCV [57] is an alternative to k-fold cross-validation that tries to keep the data distribution as similar as possible. Lopez *et al.* [58] suggest using k-fold DOB-SCV instead of k-fold cross-validation to avoid having different distributions between testing and training databases. For each execution of algorithm $a \in \mathbf{A}$ in database $d \in \mathbf{D}$, we obtain the AUC (see Section IV-A) for each fold and calculate the mean AUC over the 5 folds.

When executing the algorithms (listed in Section IV-C), some implementations crashed for some databases. We have found it difficult to assess why a specific implementation failed in one database, because some of the algorithms do not have publicly available source code, and the errors were not ones handled by the developers that could provide some useful message. To deal with this issue, we ran two experiments: the full algorithms experiment and the full classifiers experiment.

The full algorithms experiment aims to compare all algorithms, and so we removed from analysis those databases for which at least one algorithm failed. In this case, we ended up with 40 databases. By contrast, the full databases experiment aims to preserve all databases, and so we removed from analysis those algorithms that fail at least with one database. Then, we were left with 15 algorithms. This way our analysis is more robust, for as shall be seen in Section V, we can understand the effect of adding or removing algorithms or databases from our experiments.

For each experiment, we make a first comparison of the algorithms through key statistics of the AUC obtained for each database. We show these statistics with boxplots and rank the classifiers by their median AUC.

Next, for each experiment, we take each pair of algorithms $(a_i, a_j), i \neq j$ considered in the experiment and apply the Bayesian signed-rank test (see Section IV-B) for the subset of databases considered in the experiment. The test gives three probabilities as a result: the probability that a_i is practically better than a_j, in other words, the probability of a_i winning; the probability that a_j is practically better than a_i, in other words, the probability of a_j winning or a_i losing; and the probability that a_i and a_j are practically equivalent, in other words, the probability of a tie.

To identify in which databases the algorithms perform better, we also apply the Bayesian signed-rank test to subsets of the databases for each experiment. First, for each experiment, we compare the results in databases with two classes against databases with more than two classes. Therefore, we will have four subsets of databases, shown in Table 6, where we apply the test. The other comparisons are in databases with up to 20 features, against databases with more than 20 features; databases with up to 1,000 objects, against databases with more than 1,000 objects; and databases with up to two objects of the majority class for each object in the minority class, against databases with more than two objects of the majority class for each object in the minority class.

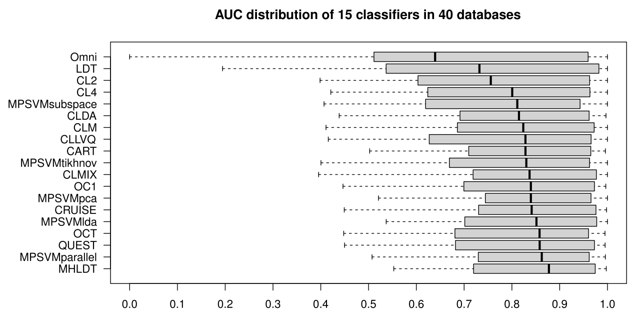

FIGURE 4. Boxplot showing the distribution of AUC of the 19 algorithms described in Section IV in 40 databases for the full algorithms experiment. The algorithms are sorted by their median AUC, with the algorithms with highest median AUC at the bottom.

V. RESULTS AND DISCUSSION

In this section, we show the results of our comparison of the 19 MDT induction algorithms. As discussed in Section IV-D, we have conducted two experiments, which we call the full algorithms experiment and the full databases experiment.

In Figure 4, we show a boxplot of the distribution of AUC for the full algorithms experiment. The algorithms are ordered according to their median AUC, with algorithms with the highest values at the bottom. The boxplot helps us to visualize the distribution of AUC for each algorithm, showing the minimum and maximum values with the whiskers (left and right small vertical lines at the edge of each dashed line), the median (bold line inside the box), and the first and third quartiles (left and right edges of the box).

Since we want to maximize AUC, we aim to build algorithms with high median AUC and low variability. Visually, small boxes and whiskers closer to the median indicate low variability. MHLDT has the highest median AUC, which may indicate good performance. From Figure 4, we notice that the median AUC of most algorithms is in the range $(0.8, 0.9)$, and there is great overlap between the boxes. Even so, a consistent difference in performance statistics of AUC (median, first quartile, third quartile, and minimum) indicates that one algorithm is performing better than the other. However, a consistent improvement of AUC of 0.01 might not be easy to notice visually; however, we will detect this consistent difference with the statistical tests. Other differences are more noticeable; for example, we know for sure that Omni obtains worse results compared to MHLDT in at least 25% of the databases because the first quartile of Omni is lower than the minimum AUC of MHLDT.

In Figure 5, we show a boxplot with the distribution of AUC for the full databases experiment. We notice that when adding 17 databases, the minimum, median, and first

TABLE 7. Bayesian signed-rank test results for the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.77	0.98	0.98	1.00	0.95	0.96	0.85	0.94	0.98	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(2) CRUISE	0.02	-	0.55	0.52	0.68	0.49	0.67	0.61	0.43	0.80	0.87	0.96	0.95	1.00	1.00	1.00	1.00	1.00	1.00
(3) MPSVMlda	0.01	0.45	-	0.58	0.57	0.53	0.47	0.56	0.67	0.56	0.85	0.80	0.87	1.00	1.00	1.00	1.00	1.00	1.00
(4) MPSVMpca	0.01	0.48	0.40	-	0.49	0.57	0.39	0.55	0.82	0.68	0.88	0.91	0.91	1.00	1.00	1.00	1.00	1.00	1.00
(5) CART-LC	0.00	0.27	0.42	0.25	-	0.28	0.37	0.42	0.71	0.64	0.78	0.87	0.94	0.99	1.00	1.00	1.00	1.00	1.00
(6) OC1	0.01	0.27	0.34	0.39	0.23	-	0.42	0.33	0.44	0.65	0.80	0.93	0.85	1.00	1.00	1.00	1.00	1.00	1.00
(7) MPSVMparallel	0.00	0.31	0.18	0.35	0.27	0.54	-	0.32	0.77	0.67	0.84	0.85	0.91	1.00	1.00	1.00	1.00	1.00	1.00
(8) OCT	0.00	0.31	0.43	0.45	0.36	0.32	0.40	-	0.62	0.49	0.58	0.83	0.90	0.99	1.00	1.00	1.00	1.00	1.00
(9) QUEST	0.00	0.02	0.17	0.18	0.29	0.40	0.22	0.31	-	0.48	0.58	0.63	0.70	0.92	0.99	1.00	1.00	1.00	1.00
(10) CLMIX	0.01	0.06	0.30	0.29	0.36	0.27	0.31	0.43	0.27	-	0.64	0.43	0.69	0.97	1.00	1.00	1.00	1.00	1.00
(11) CLDA	0.00	0.04	0.13	0.12	0.22	0.18	0.16	0.41	0.33	0.18	-	0.56	0.66	0.96	1.00	0.98	1.00	1.00	1.00
(12) CLM	0.00	0.02	0.16	0.07	0.13	0.05	0.12	0.16	0.32	0.10	0.43	-	0.50	0.89	1.00	0.99	1.00	1.00	1.00
(13) MPSVMtikhnov	0.00	0.05	0.00	0.07	0.06	0.14	0.02	0.10	0.27	0.29	0.34	0.48	-	0.93	0.99	0.98	1.00	0.99	1.00
(14) CLLVQ	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.08	0.01	0.04	0.00	0.07	-	0.75	0.80	1.00	0.98	1.00
(15) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.01	-	0.53	0.86	0.96	1.00
(16) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.02	0.20	0.47	-	0.76	0.89	0.99
(17) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.24	-	0.89	1.00
(18) LDT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.04	0.11	0.11	-	1.00
(19) Omni	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	-

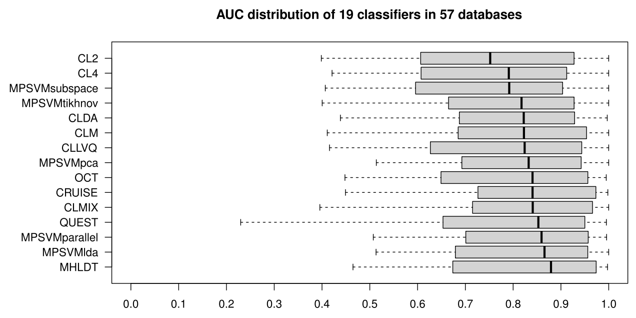

FIGURE 5. Boxplot showing the distribution of AUC of 15 of the algorithms described in Section IV in 57 databases for the full databases experiment. The algorithms are sorted by their median AUC, with the algorithms with highest median AUC at the bottom.

and third quartiles for AUC are generally lower. We also notice that the relative ranking according to the median AUC of some algorithms changes. The lowered performance of 15 classifiers might be because the 17 databases are harder to classify and furthers our motivation of giving results for all databases for the algorithms for a subset of algorithms. The full results with the AUC of each algorithm for each database are shown in Appendix B.

In the boxplots, we used the median AUC to rank the algorithms; however, the median AUC alone does not guarantee good performance, we need to consider the whole distribution of AUC. For example, by ranking the algorithms by median AUC, CRUISE is at sixth place and QUEST is at third place; however, the AUC of the first quartile is noticeably higher for CRUISE. Pairwise comparison of CRUISE and QUEST would confirm that CRUISE achieves higher AUC values most often than QUEST, which seems natural since CRUISE was published years later than QUEST by the same authors, with CRUISE preserving some successful characteristics from QUEST. Since it is difficult to assess which algorithm is better by only looking at AUC statistics, to make a fair comparison of the algorithms, we need to apply statistical tests.

A. STATISTICAL COMPARISON

In Section IV-B, we described the Bayesian signed-rank test to compare a pair of algorithms. In Table 7, we show the results of the statistical test for the full algorithms experiment. The same results for the full databases experiment are shown in Table 8. The number of each cell is the probability that the algorithm in the row is practically better than the one in the column. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing. From the probabilities of a pair of algorithms i, j winning against each other, we can obtain the probability of the algorithms being practically equivalent as $1 - p_{ij} - p_{ji}$.

For example, we see that the probability that CRUISE is practically better than QUEST is 0.43 from cell $(2, 9)$ of Table 7. From cell $(9, 2)$ of Table 7, we see that there is a probability that QUEST wins against CRUISE of 0.02. From the previous probabilities, we can obtain the probability that QUEST and CRUISE are practically equivalent as $1 - 0.43 - 0.02 = 0.55$. In 55% of cases, there is a bias towards the algorithms being practically equivalent. However, with no additional information, we can conclude that CRUISE outperforms QUEST because CRUISE still wins in 43% of cases, and QUEST only wins in 2% of the cases.

In Figure 6, we show the median probability of winning or losing of each algorithm, as well as the median AUC (color and size of points). Although we are only showing the median values, we can now visually compare multiple algorithms simultaneously, which is challenging to do from the tables. Furthermore, we will see that we can visually identify groups of top-performing algorithms that match the ranking used in Tables 7 and 8.

For the full algorithms experiment, in the top plot of Figure 6, we notice a group of top-performing algorithms with a median probability of winning higher than 0.7 and a median probability of losing smaller than 0.3, namely, MHLDT, CRUISE, MPSVMpca, MPSVMparallel, CART-LC, MPSVMlda, and OC1. The algorithms in this group are also at the top of the ranking in Table 7.

TABLE 8. Bayesian signed-rank test results for the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	0.87	0.99	0.96	1.00	0.92	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(2) CRUISE	0.05	-	0.57	0.69	0.67	0.80	0.30	0.99	0.98	0.99	0.98	1.00	1.00	1.00	1.00
(3) MPSVMlda	0.00	0.43	-	0.34	0.80	0.83	0.91	0.88	0.90	0.96	0.98	1.00	1.00	1.00	1.00
(4) MPSVMparallel	0.00	0.31	0.13	-	0.56	0.70	0.94	0.97	0.90	0.96	0.99	1.00	1.00	1.00	1.00
(5) MPSVMpca	0.00	0.33	0.17	0.21	-	0.70	0.91	0.94	0.84	0.92	0.96	1.00	1.00	1.00	1.00
(6) OCT	0.00	0.17	0.17	0.08	0.29	-	0.77	0.66	0.64	0.67	0.88	0.99	1.00	1.00	1.00
(7) QUEST	0.00	0.01	0.07	0.06	0.09	0.20	-	0.55	0.68	0.74	0.72	0.82	0.99	0.99	1.00
(8) CLM	0.00	0.01	0.11	0.03	0.05	0.31	0.44	-	0.35	0.71	0.67	0.94	1.00	1.00	1.00
(9) CLMIX	0.00	0.00	0.06	0.10	0.16	0.35	0.22	0.25	-	0.28	0.65	0.92	1.00	1.00	1.00
(10) CLDA	0.00	0.00	0.03	0.04	0.08	0.32	0.23	0.29	0.12	-	0.68	0.92	0.99	1.00	1.00
(11) MPSVMtikhnov	0.00	0.02	0.00	0.00	0.04	0.12	0.27	0.33	0.35	0.32	-	0.85	0.99	1.00	1.00
(12) CLLVQ	0.00	0.00	0.00	0.00	0.00	0.01	0.18	0.00	0.04	0.08	0.15	-	0.88	0.97	1.00
(13) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.01	0.12	-	0.51	0.88
(14) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.49	-	0.93
(15) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.00	-

We notice that CART-LC and OC1 are at positions 5 and 6 in the ranking of Table 7, which seems high. These algorithms are two of the most popular algorithms used in experimental comparisons, so we would expect most algorithms to outperform them. We even notice that, with a median probability greater than 0.6, CART-LC wins over 11 algorithms (labeled 9 to 19 in Table 7), and OC1 wins over 10 algorithms (labeled 10 - 19 in Table 7).

For the full algorithms experiment, in the bottom plot of Figure 6, we can identify a smaller group of five top-performing algorithms with a median probability of winning higher than 0.9 and a median probability of losing smaller than 0.1, namely, MHLDT, CRUISE, MPSVMparallel, MPSVMlda, and MPSVPpca. The algorithms in the group are also at the top of the ranking in Table 8. With this group of five algorithms in mind, we notice that the only difference in the ranking when considering all classifiers, for the full algorithms experiment, is that MPSVMparallel goes from rank 4 to 7.

We now want to identify common characteristics of MHLDT, CRUISE, MPSVMlda, and MPSVPpca, which are among the top-five ranked algorithms for the full algorithms experiment and the full databases experiment. The first common characteristic is that all algorithms use an analytical method to find the coefficients of the linear combination w. However, the specific procedure for finding w is different for each algorithm; MHLDT uses a multi-class version of Fisher's linear discriminant, CRUISE uses linear discriminant functions, producing K-ary splits, and all versions of Zhang's MPSVM use MPSVM to find clustering hyperplanes used to obtain w.

A second common characteristic is that the classifiers may use univariate splits in some cases. However, only MHLDT uses a feature selection method to obtain multivariate splits with few features, which may be an advantage. A third common characteristic between MHLDT and CRUISE, which are better ranked than MPSVM, is that they can work directly with multi-class problems.

The results shown have taken into account databases with a diverse number of features, objects, classes, and degrees of imbalance. However, an algorithm with low performance in a group of databases, such as high-imbalance databases, may have competitive performance in another group, such as low-imbalance databases. We now make statistical comparisons by type of database to identify top-performing algorithms in groups of databases.

B. ANALYSIS BY TYPE OF DATABASE

In this section, we will show the results of the statistical test for subsets of our databases according to the number of features, objects, classes, and degree of imbalance. Since we will have 16 comparisons, we only show the results with figures similar to 6. The results in table form can be consulted in Appendix B.

1) NUMBER OF CLASSES

First, let us compare the performance of the classifiers in a subset of databases with two classes against a subset of databases with more than two classes. In Figure 7, the upper subplots correspond to database subsets for the full algorithms experiment, while the lower subplots correspond to database subsets for the full databases experiment. The left subplots have database subsets containing only two classes, while the right subplots have database subsets containing more than two classes. We notice that the median AUC for all classifiers is lower for databases with two classes than for databases with more than two classes; this suggests that the selected two-class databases may be more difficult to classify correctly.

For both experiments, when considering a subset of databases with more than two classes, we notice that the five algorithms with the higher median probability of winning and lower median probability of losing are MHLDT, CRUISE, MPSVMlda, MPSVMparallel, and MPSVMpca. This group of algorithms matches the one obtained in the test considering all 57 databases for the full databases experiment, we can verify this in Figure 6.

For the case with only two classes, for the full databases experiment, we add QUEST to the group of five top-performing algorithms identified in Figure 6. OCT may also be considered because it has a median probability of winning slightly higher than CRUISE, but also a higher median

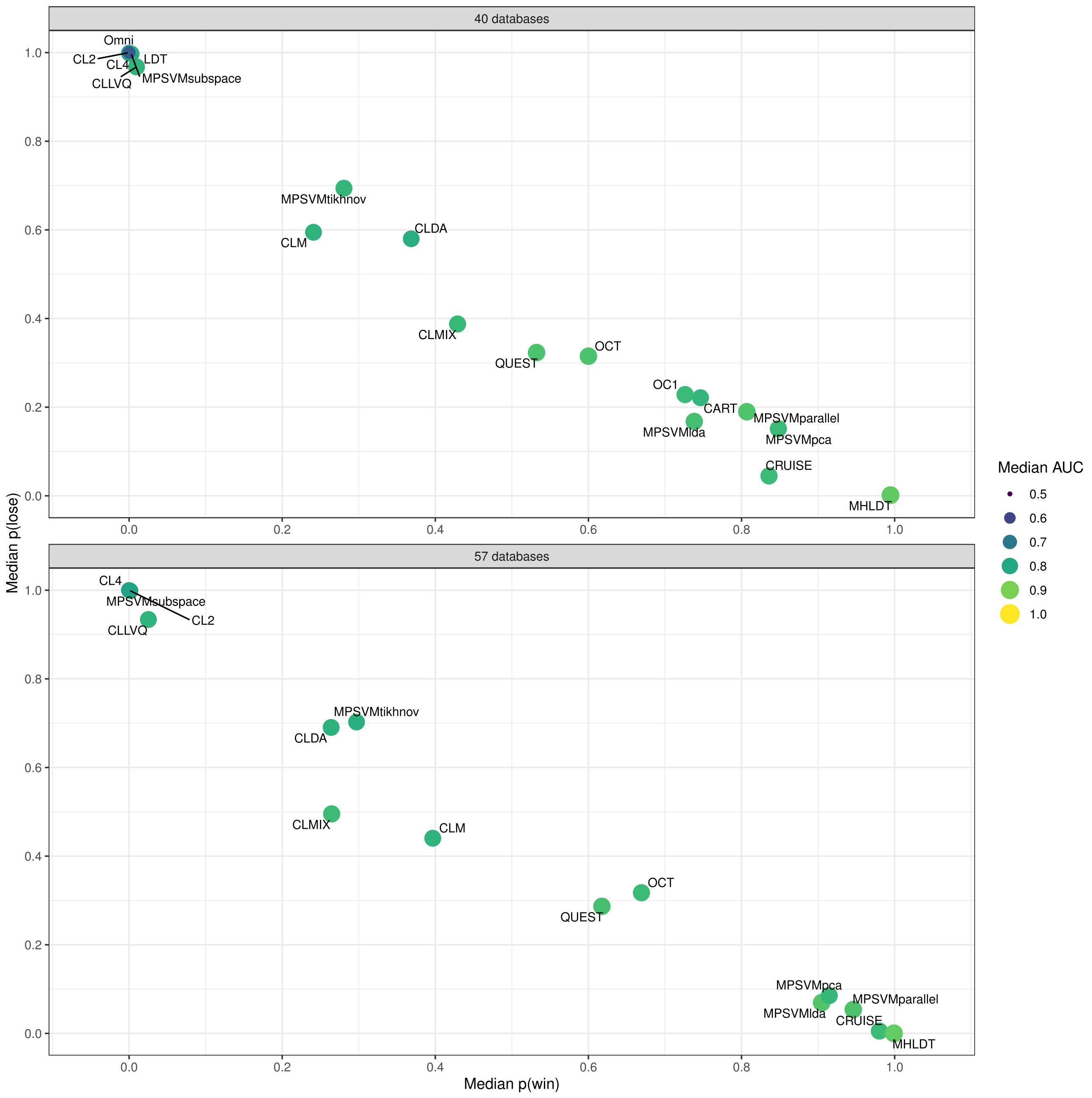

FIGURE 6. Statistical comparison of classification performance, using the Bayesian signed-rank test. The upper subplot shows the results of the full algorithms experiment, and the lower subplot shows the results of the full databases experiment. Each subplot shows the median probability of each algorithm winning and the median probability of each algorithm losing; the best performing algorithms are at the bottom right corner, while the worst performing algorithms are at the upper left corner. We also show the median AUC through the size and color of the points.

probability of losing. For the full algorithms experiment, the different versions of MPSVM have reduced performance. Now we can identify a different group of six top-performing classifiers, including MHLDT, CART-LC, OC1, QUEST, CRUISE, and OCT.

Identifying groups of algorithms that perform well in database subsets with common characteristics can help us select which algorithms to test given a specific database. For example, we would test QUEST before CRUISE for two-class databases since it has a slightly higher median probability of winning and a slightly lower median probability of losing. However, if the objective is to design a new algorithm that works well in databases with more than two classes, we would like to know what CRUISE is doing different from QUEST to achieve an improvement in classification performance.

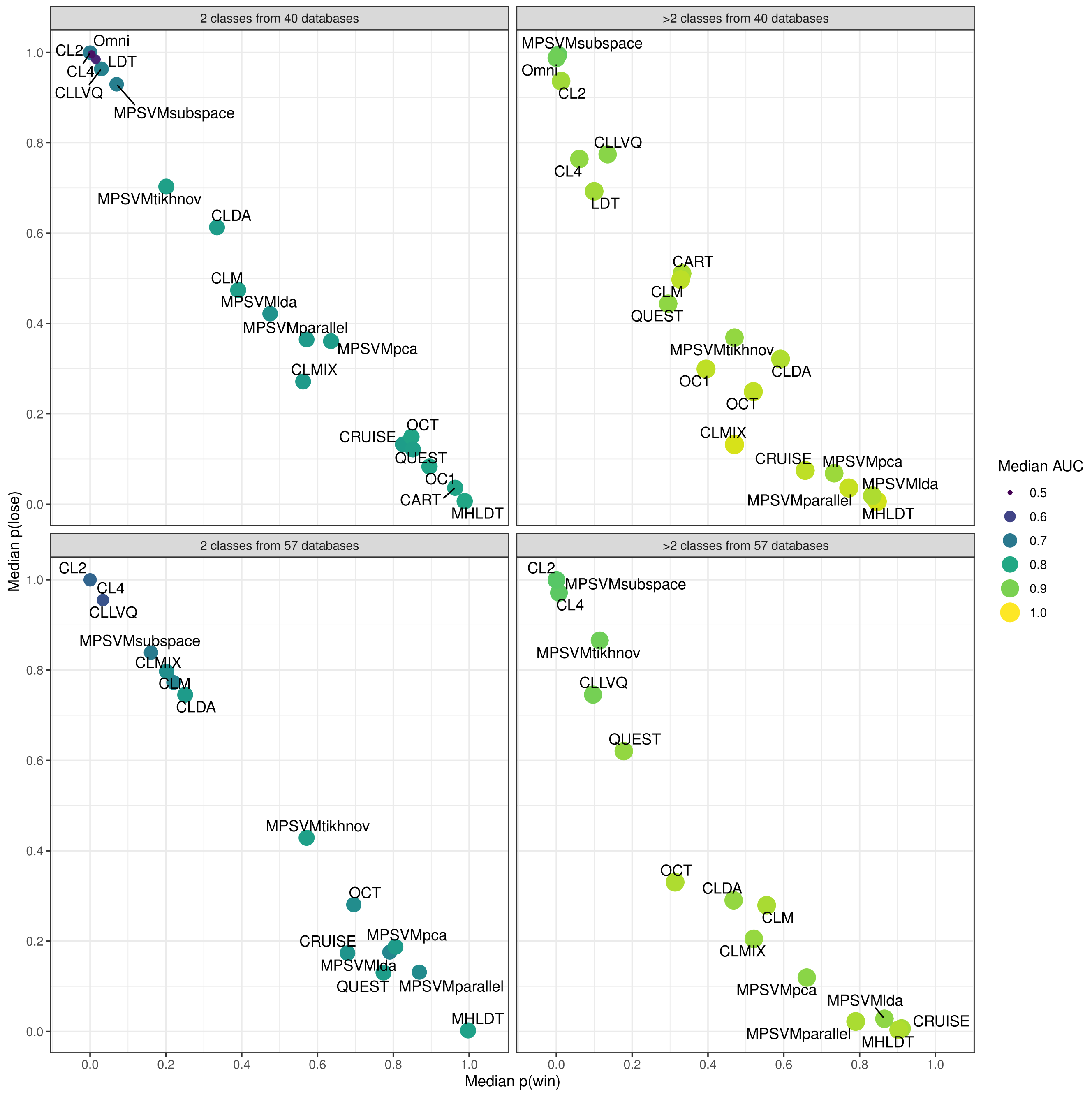

FIGURE 7. **Statistical comparison of classification performance by the number of classes, using the Bayesian signed-rank test. The upper subplots show the results for the full algorithms experiment, and the lower subplots show the results for the full databases experiment. The left subplots show the results for database subsets with two classes, and the right subplots show the results for database subsets with more than two classes. Each subplot shows the median probability of each algorithm winning and the probability of each algorithm losing; the best performing algorithms are at the bottom right corner, while the worst performing algorithms are at the upper left corner. We also show the median AUC through the size and color of the points.**

Although CART-LC and OC1 are the oldest MDTs we compare, they manage to outperform most of the algorithms for two-class databases. However, CART-LC and OC1 have reduced performance for databases with more than two classes. Considering that the top-performing algorithms are analytical algorithms, this result may suggest that iterative algorithms may have difficulties finding good splits for databases with more than two classes.

2) NUMBER OF FEATURES

In Figure 8, the upper subplots correspond to database subsets for the full algorithms experiment, while the lower subplots correspond to database subsets for the full databases experiment. The left subplots have database subsets with up to 20 features, while the right subplots have database subsets containing more than 20 features. We notice that the median AUC for all classifiers is lower for databases with up to

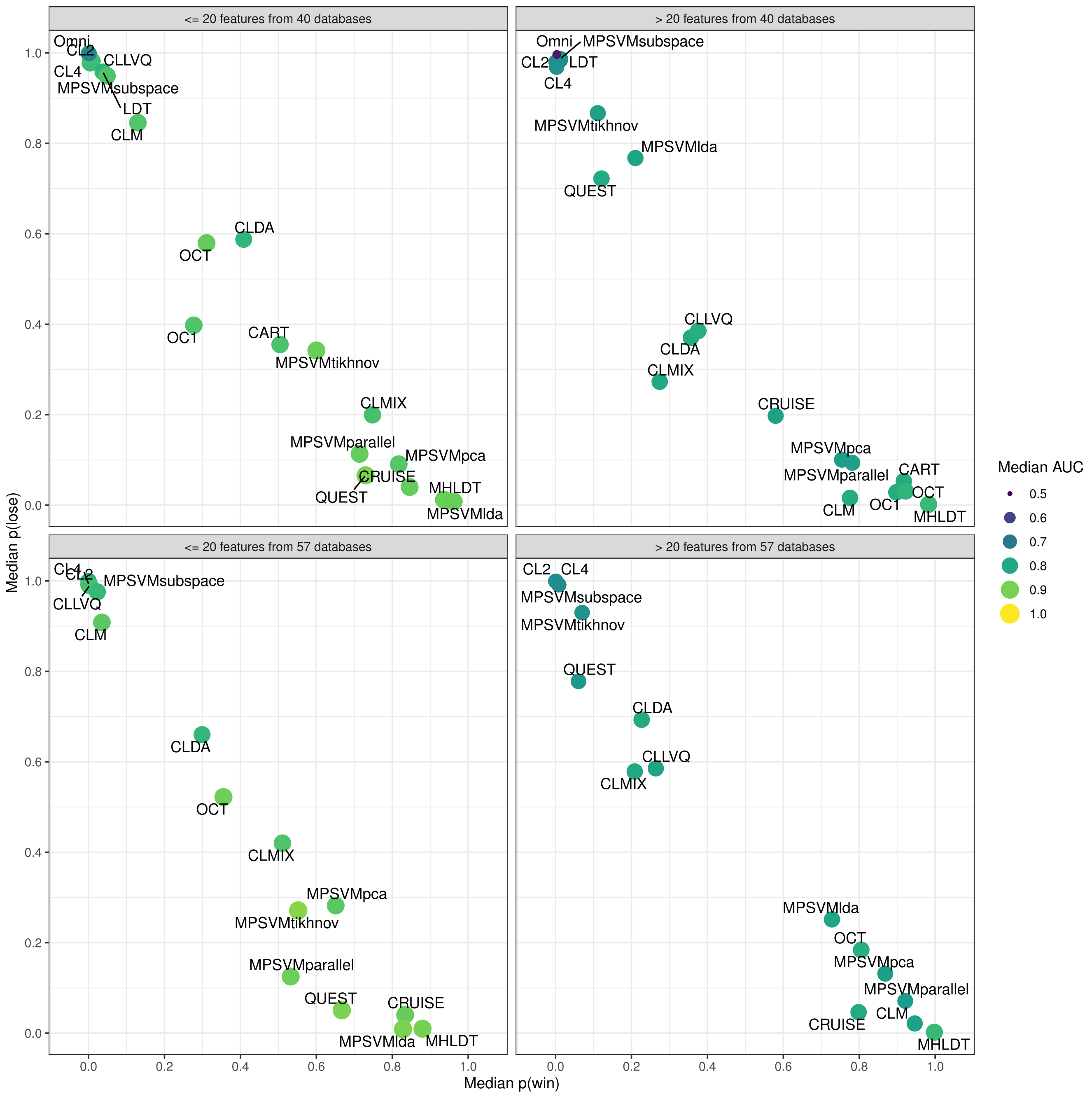

FIGURE 8. Statistical comparison of classification performance by the number of features, using the Bayesian signed-rank test. The upper subplots show the results for the full algorithms experiment, and the lower subplots show the results for the full databases experiment. The left subplots show the results for database subsets with up to 20 features, and the right subplots show the results for database subsets with more than 20 features. Each subplot shows the median probability of each algorithm winning and the probability of each algorithm losing; the best performing algorithms are at the bottom right corner, while the worst performing algorithms are at the upper left corner. We also show the median AUC through the size and color of the points.

20 features than for databases with more than 20 features; this suggests that the problems with more features are more difficult to classify correctly.

For the case with up to 20 features, for both experiments, we identify a small group of three top-performing algorithms: MHLDT, MPSVMlda, and CRUISE.

For the case with more than 20 features, the group of three best performing classifiers changes. We can still identify CRUISE and MHLDT in a group of five top-performing algorithms for the full databases experiment, but we only identify MHLDT in the group of five top-performing algorithms for the full algorithms experiment.

The top-performing algorithms for databases with few features are all analytical algorithms. However, we do not notice a clear pattern for the top-performing algorithms

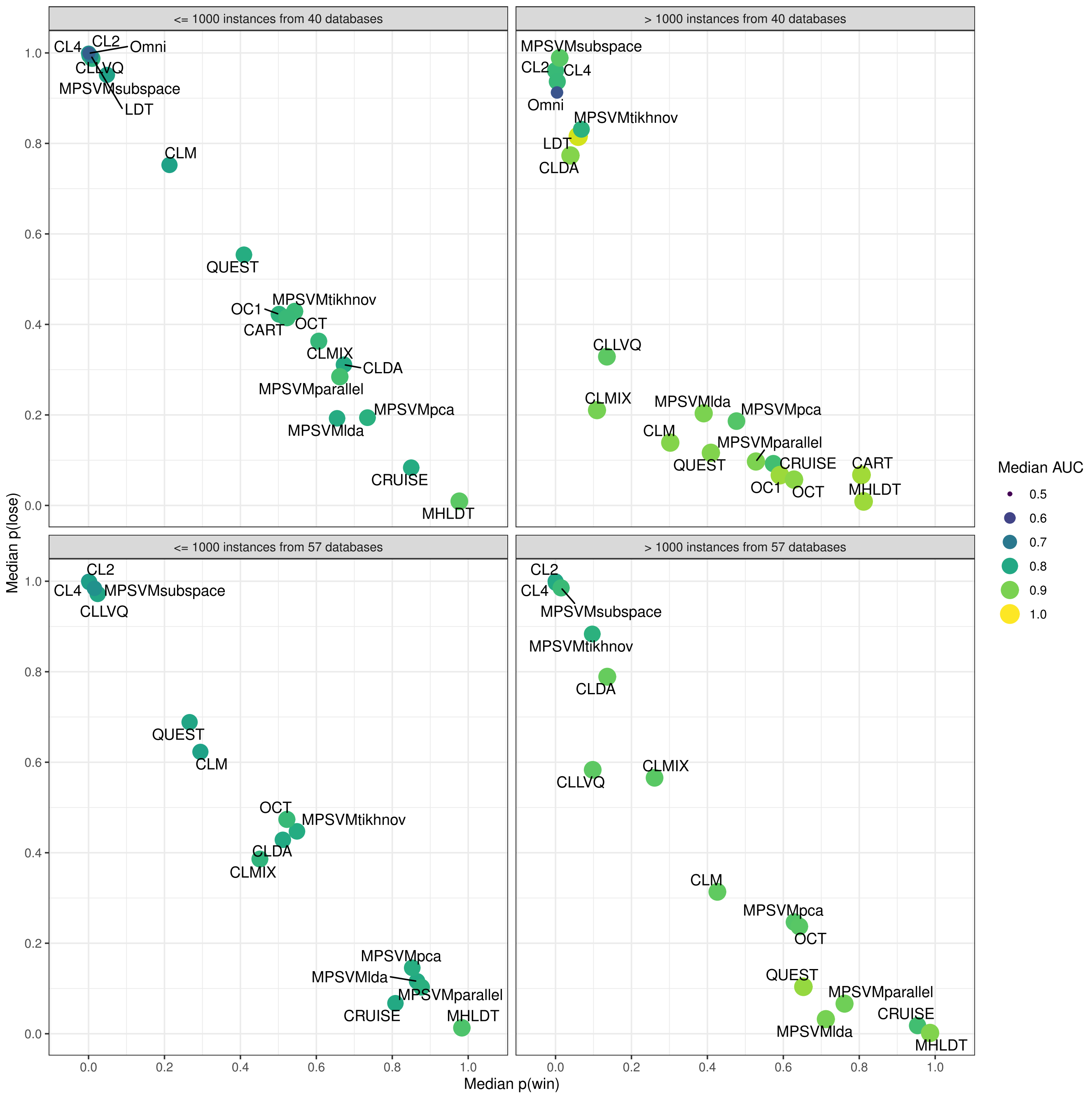

FIGURE 9. Statistical comparison of classification performance by the number of objects, using the Bayesian signed-rank test. The upper subplots show the results for the full algorithms experiment, and the lower subplots show the results for the full databases experiment. The left subplots show the results for database subsets with up to 1,000 objects, and the right subplots show the results for database subsets with more than 1,000 features. Each subplot shows the median probability of each algorithm winning and the probability of each algorithm losing; the best performing algorithms are at the bottom right corner, while the worst performing algorithms are at the upper left corner. We also show the median AUC through the size and color of the points

in databases with many features. For the full algorithms experiment, the top-performing algorithms include an analytical algorithm with feature selection (MHLDT), an iterative algorithm with feature selection (CART-LC), an iterative algorithm without feature selection (OC1), and a tree optimization algorithm (OCT).

3) NUMBER OF OBJECTS

In Figure 9, the upper subplots correspond to database subsets for the full algorithms experiment, while the lower subplots correspond to database subsets for the full databases experiment. The left subplots have database subsets with up to 1,000 objects, while the right subplots have database

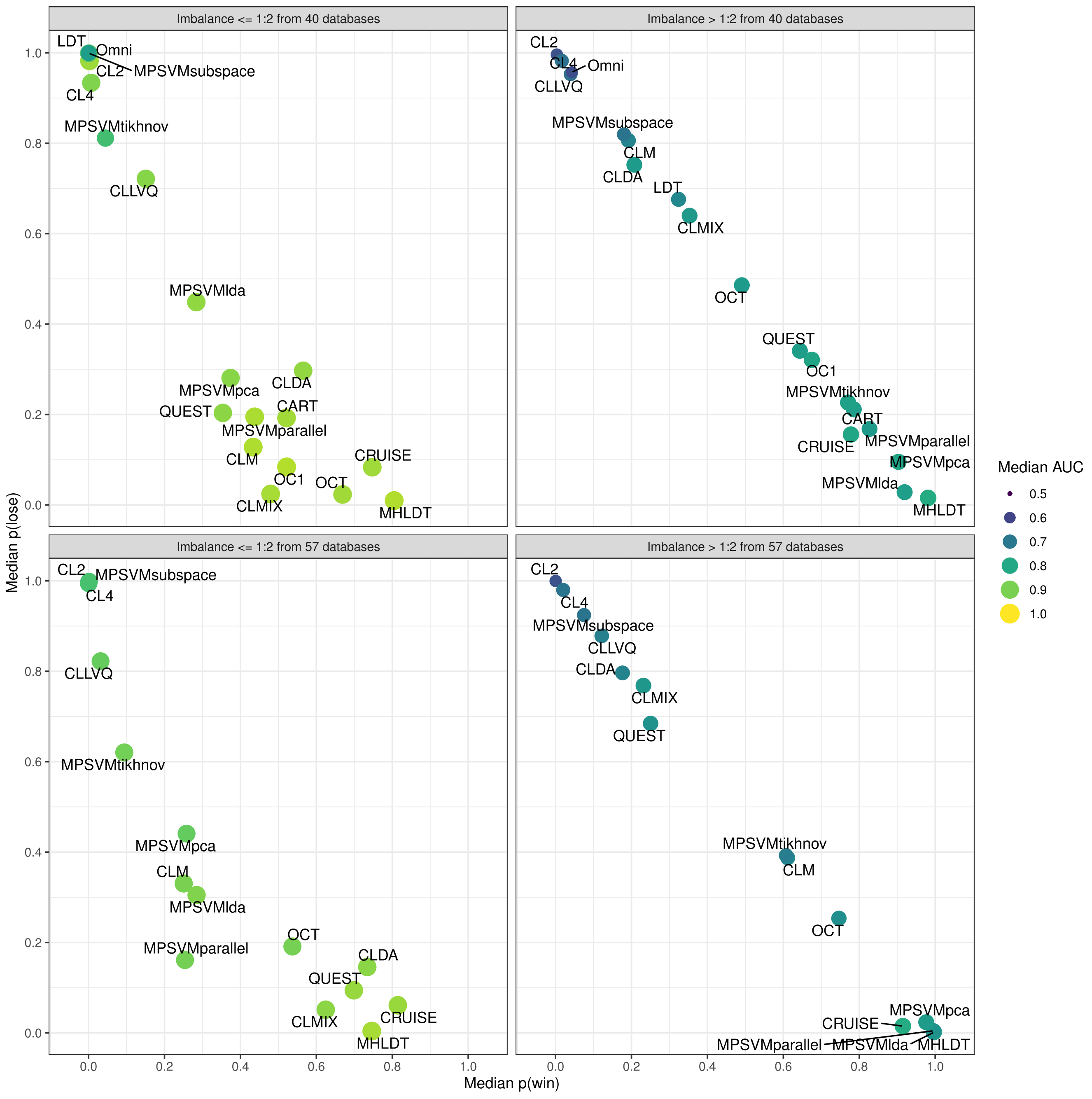

FIGURE 10. **Statistical comparison of classification performance by the degree of imbalance, using the Bayesian signed-rank test. The upper subplots show the results for the full algorithms experiment, and the lower subplots show the results for the full databases experiment. The left subplots show the results for database subsets with up to 2 objects of the majority class for each object of the minority class, and the right subplots show the results for database subsets with more than 2 objects of the majority class for each object of the minority class. Each subplot shows the median probability of each algorithm winning and the probability of each algorithm losing; the best performing algorithms are at the bottom right corner, while the worst performing algorithms are at the upper left corner. We also show the median AUC through the size and color of the points**

subsets containing more than 1,000 objects. We notice that the median AUC for all classifiers is lower for databases with up to 1,000 objects than for databases with more than 1,000 objects; this suggests that the problems with fewer objects are more difficult to classify correctly.

For the case with up to 1,000 objects, we notice that the five top-performing algorithms for the full databases experiment are again MHLDT, CRUISE, MPSVMPparallel, MPSVMPlda, and MPSVPpca. This group of algorithms can also be identified for the full algorithms experiment, which considers all algorithms.

For the case with more than 1,000 objects, for the full databases experiment, the group of five top-performing algorithms includes QUEST instead of MPSVMpca. For the full algorithms experiment, the group of top-performing algorithms changes, now including MHLDT, CART-LC, OCT, OC1, and CRUISE.

The top-performing algorithms for databases with few objects are again analytical algorithms. For the databases with many objects, we have similar results to the databases with many features; the top-performing algorithms MHLDT, CART-LC, OCT, and OC1, follow different approaches to generate splits.

4) IMBALANCE

In Figure 10, the upper subplots correspond to database subsets for the full algorithms experiment, while the lower subplots correspond to database subsets for the full databases experiment. The left subplots have database subsets with low imbalance, which we consider as databases with up to 2 objects of the majority class for each object of the minority class. The right subplots have database subsets with high imbalance, which we consider as databases with more than 2 objects of the majority class for each object of the minority class. We notice that the median AUC for all classifiers is lower for imbalanced databases than for balanced databases; this suggests that problems with high imbalance are more difficult to classify correctly.

For the case of balanced databases, for the full databases experiment, the five top-performing algorithms are CRUISE, MHLDT, CLDA, QUEST, and CLMIX. For the full algorithms experiment, the group changes, now including MHLDT, CRUISE, OCT, CLMIX, and OC1.

For the case of imbalanced databases, for both experiments, the five top-performing algorithms are again MHLDT, CRUISE, MPSVMPparallel, MPSVMPlda, and MPSVPpca.

The top-performing algorithms for imbalanced databases are analytical algorithms. However, the top-performing algorithms for balanced databases, including MHLDT, CRUISE, OCT, and OC1, follow different approaches for split generation.

VI. CONCLUSION

In this paper, we identified a gap in Multivariate Decision Tree (MDT) literature. There are no surveys about MDTs, and recent papers introducing MDTS suffer from one or two main shortcomings in their comparison of previous work: authors do not compare their algorithm with relevant rival techniques, or they do so but not in enough databases, and hence results are insufficient for statistically validating the underlying hypothesis.

Our goal is to evaluate the relative merit of published MDT induction algorithms to identify how they compare to one another. By doing so, we aim to fill a gap in MDT literature, and we hope that our findings help the community to select an MDT to use in a particular context.

To accomplish our goal, first, we conducted a survey of 37 relevant MDT induction algorithms. Then, we evaluate the classification performance of 19 general-purpose MDT induction algorithms in 57 databases and make a statistical comparison.

Our conclusions are stronger than any other found in the literature since we compare almost four times the number of algorithms as the most thorough MDT study, which compared 5 algorithms in 20 databases [11]. We also compare the algorithms in 4 more databases than the reviewed study with the largest number of databases [14].

Our main contributions in this paper are: (1) we provide a sound and extensive review of MDT induction algorithms; (2) we provide the most extensive MDT comparison to date, supporting our results through statistical tests; (3) we identify groups of top-performing algorithms for all databases and subsets of databases with common characteristics.

We provide the full results of the Bayesian signed-rank tests for each pair of algorithms, which can be used to find if there is a bias towards either algorithm winning or towards a tie. To summarize the results of the Bayesian signed-rank tests and compare the overall performance of the algorithms, we made plots with the median probability of an algorithm winning or losing against other algorithms. With these plots, we were able to identify groups of top-performing algorithms in all databases and in subsets of databases according to the number of classes, number of features, number of objects, and degree of class imbalance.

The two top-performing algorithms, when considering all databases, are CRUISE [18] and MHLDT [20]. Both algorithms are analytical algorithms that can work directly with multi-class problems. However, we notice that MHLDT often outperforms CRUISE and one differentiating characteristic is that MHLDT uses feature selection. Next in the ranking are some versions of Zhang's MPSVM [2], which is also an analytical algorithm, but lacks feature selection and cannot work directly with multi-class problems. Given these results, we would encourage exploring improvements in analytical MDTs, taking into account feature selection, and taking care of how to work with multi-class databases.

For specific types of databases, we found that MHLDT and CRUISE are often among the top-performing algorithms. We also found that CART [34] and OC1 [9] outperform many algorithms in some subsets of databases, even though they are the oldest algorithms in the comparison and new algorithms are often compared against them. Specifically, CART and OC1 are among the top-performing algorithms in two-class databases, in databases with more than 20 features, and in databases with more than 1,000 objects. This result highlights the importance of comparing new MDT algorithms against previous algorithms in a sufficient number of diverse databases to allow making a statistical comparison.

A. CHALLENGES FOR MDTs AND FUTURE WORK

We have identified the following challenges for MDTs that can motivate future work:

TABLE 9. Details about the 57 databases used in the experimental comparison. The top 40 databases work with all algorithms and are used for the full algorithms experiment. For each dataset, we show the number of objects, number of features (without the class), number of classes. The column Imbalance shows the number of objects of the majority class for each object of the minority class. References for the databases with a citation request are included.

Database	#Objects	#Features	#Classes	Imbalance
Balance Scale	625	4	3	5.88
QSAR biodegradation [64]	1055	41	2	1.96
Breast Tissue	106	9	6	1.57
Climate Model Simulation Crashes [65]	540	20	2	10.74
Cloud	108	4	4	1.33
CNAE-9	1080	856	9	1.00
Vertebral Column (two class)	310	6	2	2.10
Vertebral Column (three class)	310	6	3	2.50
dataset3d [20]	120	3	2	1.00
Pima Indians Diabetes	768	8	2	1.87
Ecoli	336	7	8	71.50
Glass Identification	214	9	6	8.44
Haberman's Survival	306	3	2	2.78
Ionosphere	351	34	2	1.79
Iris	150	4	3	1.00
User Knowledge Modeling [66]	403	5	4	2.58
Letter Recognition	20000	16	26	1.11
Liver Disorders	345	6	2	1.38
LSVT Voice Rehabilitation [67]	126	310	2	2.00
Madelon [68]	2600	500	2	1.00
Libras Movement	360	90	15	1.00
Optical Recognition of Handwritten Digits	5620	64	10	1.03
Parkinson Speech Dataset with Multiple Types of Sound Recordings [69]	1208	26	2	1.32
Parkinsons [70]	195	22	2	3.06
Pen-Based Recognition of Handwritten Digits	10992	16	10	1.08
seeds	210	7	3	1.00
Image Segmentation	2310	19	7	1.00
Connectionist Bench (Sonar, Mines vs. Rocks)	208	60	2	1.14
Spambase	4601	57	2	1.54
SPECTF Heart	267	44	2	3.85
Low Resolution Spectrometer	531	100	4	3.63
Statlog (Image Segmentation)	2310	19	7	1.00
Statlog (Vehicle Silhouettes) [71]	846	18	4	1.10
Connectionist Bench (Vowel Recognition - Deterding Data) Data Set	990	10	11	1.00
Breast Cancer Wisconsin (Diagnostic)	569	30	2	1.68
Wholesale customers [72]	440	7	2	2.10
Wilt [73]	4839	5	2	17.54
Wine	178	13	3	1.48
Wine Quality (red) [74]	1599	11	6	68.10
Yeast	1484	8	10	92.60
Quality Assessment of Digital Colposcopies (Green) [75]	98	62	2	2.16
Quality Assessment of Digital Colposcopies (Hinselman) [75]	97	62	2	5.47
Quality Assessment of Digital Colposcopies (Schiller) [75]	92	62	2	2.68
Cardiotocography Data Set	2126	41	3	9.40
Geographical Original of Music (default features) [76]	1059	68	33	6.27
Geographical Original of Music (default + chromatic features) [76]	1059	116	33	6.27
Steel Plates Faults [77]	1941	27	7	9.73
Forest type mapping [78]	523	27	4	2.35
Gesture Phase Segmentation [79]	9873	32	5	2.96
Hill-Valley (without noise)	1212	100	2	1.02
HTRU2 [80]	17898	8	2	9.92
MAGIC Gamma Telescope	19020	10	2	1.84
Ozone Level Detection (eight hour)	2534	72	2	14.84
Urban Land Cover [81]	675	147	9	4.21
Waveform Database Generator (Version 1)	5000	21	3	1.03
Waveform Database Generator (Version 2)	5000	40	3	1.02
Wireless Indoor Localization [82]	2000	7	4	1.00

TABLE 10. AUC of the 19 algorithms in the 57 databases. The first 40 databases correspond to the subset of databases used in the full algorithms experiment. The full databases experiment does not include the last four algorithms, since we do not have their AUC on 17 databases. The names are abbreviated, but the databases are in the same order of Table 9.

Database	MHLDT	CRUISE	MPSVMlda	CLMIX	CLDA	QUEST	MPSVMparallel	MPSVMpca	CLM	MPSVMtikhnov	OCT	CLLVQ	MPSVMsubspace	CL4	CL2	OC1	CART	LDT	Omni
balance-scale	0.88	0.88	0.91	0.86	0.78	0.92	0.91	0.82	0.73	0.93	0.62	0.68	0.89	0.69	0.63	0.87	0.69	0.78	0.61
biodeg	0.83	0.83	0.81	0.83	0.82	0.83	0.78	0.78	0.83	0.80	0.81	0.81	0.72	0.76	0.73	0.80	0.80	0.41	0.39
breast-tissue	0.84	0.78	0.80	0.82	0.84	0.80	0.87	0.87	0.79	0.82	0.84	0.86	0.56	0.84	0.75	0.82	0.77	0.71	0.68
climate	0.81	0.76	0.73	0.77	0.77	0.81	0.74	0.69	0.54	0.71	0.58	0.54	0.69	0.54	0.50	0.60	0.75	0.19	0.00
cloud	0.61	0.55	0.62	0.54	0.62	0.48	0.62	0.59	0.49	0.57	0.58	0.44	0.57	0.47	0.54	0.49	0.58	0.27	0.38
cnae-9_fixed	0.98	0.76	0.96	0.97	0.96	0.91	0.98	0.98	0.99	0.71	0.96	0.99	0.87	0.96	0.96	0.98	0.98	0.97	0.97
column_2c	0.78	0.81	0.80	0.77	0.83	0.79	0.77	0.81	0.77	0.80	0.77	0.76	0.73	0.79	0.69	0.79	0.80	0.62	0.62
column_3c	0.90	0.88	0.87	0.88	0.88	0.89	0.85	0.89	0.88	0.85	0.89	0.84	0.84	0.77	0.81	0.88	0.84	0.87	0.80
dataset3d	0.97	0.97	1.00	1.00	0.78	0.99	0.95	1.00	1.00	1.00	0.96	1.00	1.00	1.00	1.00	0.98	0.98	0.72	0.51
diabetes	0.68	0.73	0.65	0.72	0.74	0.72	0.68	0.67	0.72	0.67	0.69	0.59	0.71	0.57	0.58	0.69	0.71	0.66	0.78
ecoli	0.65	0.74	0.84	0.84	0.80	0.69	0.74	0.83	0.88	0.86	0.65	0.74	0.78	0.85	0.68	0.52	0.62	0.73	0.55
glass	0.87	0.69	0.77	0.62	0.67	0.68	0.77	0.80	0.61	0.71	0.84	0.66	0.49	0.71	0.79	0.69	0.80	0.53	0.66
haberman	0.55	0.54	0.60	0.61	0.50	0.51	0.55	0.60	0.57	0.53	0.55	0.50	0.58	0.50	0.51	0.60	0.58	1.00	1.00
ionosphere	0.89	0.87	0.81	0.83	0.80	0.86	0.86	0.85	0.82	0.78	0.88	0.82	0.79	0.81	0.74	0.81	0.84	0.58	0.60
iris	0.97	0.99	0.98	0.97	0.98	0.98	0.98	0.97	0.97	0.98	0.98	0.96	0.98	0.97	0.97	0.98	0.97	0.98	0.98
knowledge	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.96	0.95	0.98	0.97	0.88	0.98	0.89	0.93	0.99	0.98	0.98	0.96
letter	0.99	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.95	0.99	0.99	0.99	0.99	0.99	0.99
liver-disorders	0.67	0.68	0.64	0.65	0.66	0.65	0.62	0.67	0.60	0.61	0.65	0.61	0.65	0.53	0.55	0.68	0.68	0.32	0.33
lsvt	0.67	0.67	0.72	0.78	0.78	0.53	0.67	0.68	0.79	0.54	0.74	0.68	0.55	0.67	0.66	0.71	0.61	0.39	0.15
madelon	0.66	0.56	0.54	0.58	0.53	0.56	0.77	0.73	0.60	0.56	0.82	0.57	0.56	0.50	0.51	0.79	0.78	0.51	0.51
movement	0.97	0.97	0.91	0.98	0.98	0.85	0.96	0.97	0.97	0.98	0.96	0.97	0.88	0.96	0.97	0.97	0.95	0.87	0.82
optdigits	0.99	1.00	0.98	1.00	1.00	0.99	0.99	0.99	0.99	0.82	0.99	0.99	0.95	0.99	0.99	0.99	0.99	0.99	0.98
parkinson_sound	0.66	0.65	0.63	0.63	0.66	0.64	0.64	0.63	0.64	0.63	0.67	0.63	0.59	0.59	0.61	0.65	0.68	0.36	0.41
parkinsons	0.85	0.80	0.78	0.74	0.81	0.78	0.79	0.77	0.79	0.78	0.84	0.83	0.68	0.76	0.76	0.79	0.81	0.73	0.65
pendigits	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.90	0.99	0.99	1.00	1.00	1.00	0.99
seeds	0.96	0.98	0.98	0.98	0.98	0.97	0.96	0.94	0.95	0.96	0.97	0.95	0.96	0.96	0.93	0.96	0.97	0.97	0.96
segment	0.99	0.99	0.99	0.99	0.96	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.96	0.99	0.98	0.99	0.99	0.99	0.99
sonar	0.77	0.73	0.67	0.72	0.70	0.75	0.72	0.78	0.79	0.84	0.72	0.77	0.79	0.74	0.72	0.73	0.74	0.54	0.48
spambase	0.93	0.92	0.90	0.90	0.91	0.93	0.90	0.86	0.91	0.85	0.91	0.88	0.84	0.83	0.84	0.93	0.93	0.56	0.53
spectf	0.67	0.50	0.57	0.50	0.50	0.50	0.59	0.62	0.52	0.57	0.51	0.51	0.53	0.50	0.50	0.57	0.58	1.00	0.92
spectrometer	0.78	0.77	0.68	0.59	0.57	0.48	0.75	0.76	0.68	0.67	0.80	0.58	0.69	0.58	0.58	0.82	0.78	0.66	0.54
statlog_segment	0.99	0.99	0.99	0.99	0.92	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.96	0.99	0.98	0.99	0.96	0.99	0.99
vehicle	0.91	0.93	0.92	0.91	0.92	0.91	0.89	0.88	0.87	0.91	0.88	0.85	0.83	0.86	0.83	0.86	0.88	0.89	0.84
vowel	0.98	0.99	0.98	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98	0.97	0.96	0.98	0.98	0.98	0.98	0.97	0.96
wdbc	0.94	0.95	0.96	0.93	0.96	0.97	0.94	0.93	0.94	0.93	0.93	0.91	0.91	0.91	0.93	0.96	0.93	0.46	0.42
wholesale	0.88	0.89	0.87	0.84	0.83	0.90	0.87	0.79	0.86	0.86	0.91	0.87	0.89	0.85	0.80	0.92	0.91	0.55	0.60
wilt	0.91	0.83	0.90	0.78	0.70	0.90	0.90	0.87	0.69	0.92	0.90	0.63	0.94	0.66	0.60	0.85	0.87	1.00	0.63
wine	0.98	0.99	0.99	0.98	0.98	0.97	0.97	0.91	0.98	0.97	0.95	0.99	0.96	0.95	0.98	0.94	0.92	0.99	0.94
winequality_red	0.56	0.45	0.60	0.40	0.44	0.45	0.51	0.52	0.41	0.54	0.45	0.42	0.41	0.42	0.40	0.45	0.50	0.39	0.44
yeast	0.76	0.85	0.74	0.76	0.69	0.76	0.70	0.77	0.82	0.40	0.65	0.82	0.53	0.68	0.61	0.72	0.71	0.78	0.54
colposcopy_green	0.64	0.50	0.59	0.51	0.51	0.50	0.70	0.69	0.66	0.66	0.60	0.56	0.65	0.61	0.61				
colposcopy_hinselman	0.57	0.50	0.51	0.50	0.50	0.50	0.55	0.59	0.50	0.47	0.50	0.49	0.46	0.49	0.49				
colposcopy_schiller	0.46	0.50	0.62	0.47	0.47	0.50	0.56	0.51	0.49	0.55	0.48	0.61	0.48	0.51	0.49				
ctg	0.99	0.98	0.98	0.97	0.94	0.98	0.99	0.96	0.97	0.82	0.98	0.97	0.96	0.86	0.88				
default_features_105	0.64	0.84	0.68	0.72	0.72	0.32	0.63	0.62	0.77	0.58	0.63	0.72	0.60	0.69	0.65				
default_plus_chromat	0.64	0.81	0.68	0.72	0.70	0.23	0.69	0.62	0.72	0.67	0.64	0.66	0.56	0.70	0.55				
faults	0.93	0.94	0.91	0.86	0.88	0.93	0.94	0.92	0.93	0.89	0.94	0.91	0.87	0.91	0.89				
forest_type_mapping	0.95	0.95	0.94	0.94	0.93	0.95	0.93	0.91	0.95	0.93	0.94	0.94	0.86	0.91	0.88				
gesture_phaseaset	0.78	0.71	0.75	0.61	0.61	0.71	0.75	0.76	0.73	0.66	0.75	0.72	0.75	0.64	0.62				
hill_valley_without	1.00	0.79	0.88	0.88	0.88	0.94	0.63	0.79	0.59	0.89	0.60	0.56	0.83	0.60	0.61				
htru_2	0.91	0.92	0.90	0.87	0.91	0.93	0.91	0.90	0.88	0.91	0.92	0.88	0.92	0.89	0.87				
magic04	0.81	0.79	0.79	0.75	0.78	0.82	0.80	0.78	0.75	0.81	0.83	0.77	0.81	0.72	0.72				
ozone	0.65	0.50	0.63	0.50	0.50	0.50	0.62	0.62	0.55	0.58	0.52	0.52	0.54	0.50	0.50				
urban_land_cover	0.97	0.97	0.95	0.93	0.93	0.75	0.97	0.96	0.97	0.73	0.97	0.93	0.73	0.91	0.85				
waveform-1	0.90	0.92	0.90	0.92	0.91	0.92	0.89	0.89	0.89	0.90	0.87	0.88	0.90	0.85	0.83				
waveform-2	0.89	0.92	0.90	0.91	0.92	0.92	0.88	0.87	0.89	0.89	0.87	0.88	0.90	0.80	0.78				
wifi_localization	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.60	0.98	0.98				

TABLE 11. Bayesian signed-rank test results for databases with two classes from the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.82	0.72	0.73	0.80	0.92	0.99	0.99	0.97	0.97	0.99	0.97	1.00	1.00	1.00	1.00	1.00	0.99	1.00
(2) CART	0.18	-	0.32	0.69	0.73	0.81	0.97	0.95	0.95	0.89	0.96	0.96	0.99	1.00	1.00	1.00	1.00	0.99	1.00
(3) OC1	0.27	0.06	-	0.55	0.69	0.65	0.91	0.80	0.87	0.74	0.88	0.95	0.92	0.98	1.00	1.00	1.00	0.99	1.00
(4) OCT	0.13	0.16	0.38	-	0.57	0.55	0.89	0.75	0.72	0.72	0.84	0.86	0.88	0.93	1.00	1.00	1.00	0.98	1.00
(5) CRUISE	0.01	0.12	0.22	0.36	-	0.24	0.81	0.77	0.85	0.62	0.91	0.73	0.92	0.97	0.99	1.00	1.00	0.98	1.00
(6) QUEST	0.03	0.17	0.31	0.38	0.15	-	0.76	0.84	0.90	0.55	0.86	0.68	0.90	0.95	0.96	1.00	1.00	0.98	0.99
(7) MPSVMparallel	0.01	0.02	0.09	0.06	0.16	0.24	-	0.39	0.50	0.52	0.68	0.62	0.88	0.97	0.97	1.00	1.00	0.99	1.00
(8) MPSVMlda	0.01	0.04	0.06	0.25	0.23	0.14	0.39	-	0.27	0.57	0.42	0.53	0.63	0.93	0.97	1.00	1.00	0.99	1.00
(9) CLMIX	0.03	0.04	0.13	0.24	0.11	0.07	0.49	0.71	-	0.40	0.63	0.42	0.73	0.94	0.97	1.00	1.00	0.99	1.00
(10) MPSVMpca	0.03	0.09	0.24	0.28	0.38	0.45	0.46	0.40	0.59	-	0.68	0.72	0.76	0.98	0.98	1.00	1.00	0.99	1.00
(11) CLDA	0.01	0.04	0.12	0.16	0.05	0.10	0.32	0.57	0.27	0.32	-	0.35	0.66	0.74	0.88	0.99	1.00	0.97	0.99
(12) CLM	0.03	0.04	0.05	0.14	0.19	0.25	0.34	0.44	0.23	0.28	0.59	-	0.67	0.87	0.91	1.00	1.00	0.97	1.00
(13) MPSVMtikhnov	0.00	0.01	0.08	0.12	0.08	0.08	0.07	0.04	0.26	0.14	0.34	0.32	-	0.79	0.84	1.00	1.00	0.99	0.99
(14) MPSVMsubspace	0.00	0.00	0.02	0.07	0.03	0.05	0.03	0.07	0.06	0.02	0.26	0.13	0.16	-	0.56	0.92	0.99	0.98	0.99
(15) CLLVQ	0.00	0.00	0.00	0.00	0.01	0.03	0.03	0.03	0.03	0.01	0.12	0.00	0.14	0.44	-	0.98	1.00	0.93	0.99
(16) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.08	0.00	-	0.83	0.92	0.99
(17) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	-	0.91	0.99
(18) LDT	0.01	0.01	0.01	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.03	0.03	0.01	0.02	0.07	0.08	0.09	-	0.98
(19) Omni	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.02	-

1) We have noticed that most MDT induction algorithms, including recent ones, focus on linear multivariate splits. Using non-linear multivariate splits, in addition to univariate and linear multivariate splits, may result in improved classification performance.
2) UDTs have been successfully applied to unsupervised classification (clustering) with the UD3 [59] and eUD3.5 [60] algorithms with the advantages of fast training times, and that clusters can be explained. When clusters cannot be easily divided by univariate splits, we expect that MDTs provide better performance than UDTs.
3) UDTs have also been successfully applied to semi-supervised learning. Tanha *et al.* [61], show how a UDT can be effectively used for self-training. Adapting an MDT for the self-training problem may increase classification performance.
4) The problem of identifying tree similarities has been studied for UDTs. For example, DTreeSim [62] extracts sequences by traversing trees from the root to each leaf. Frequent subsequences are identified with

TABLE 12. Bayesian signed-rank test results for databases with more than two classes from the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.57	0.78	0.45	0.50	0.33	0.74	0.78	0.91	0.83	0.73	0.86	0.99	0.93	0.98	0.89	1.00	1.00	1.00
(2) MPSVMlda	0.12	-	0.51	0.35	0.59	0.40	0.83	0.73	0.84	0.85	0.68	0.76	0.96	0.98	0.98	0.86	0.99	1.00	1.00
(3) MPSVMpca	0.02	0.40	-	0.17	0.57	0.46	0.92	0.48	0.81	0.66	0.89	0.64	0.52	0.94	0.97	0.93	0.99	0.99	1.00
(4) MPSVMparallel	0.04	0.10	0.10	-	0.53	0.63	0.78	0.64	0.80	0.76	0.89	0.79	0.49	0.86	0.97	0.73	0.99	1.00	1.00
(5) CRUISE	0.14	0.12	0.34	0.38	-	0.36	0.53	0.60	0.71	0.54	0.57	0.95	0.89	0.98	0.97	0.97	0.99	1.00	1.00
(6) CLMIX	0.01	0.03	0.19	0.21	0.09	-	0.49	0.48	0.35	0.35	0.39	0.46	0.84	0.78	0.74	0.54	0.77	0.99	1.00
(7) CLDA	0.01	0.01	0.06	0.09	0.18	0.13	-	0.66	0.54	0.33	0.55	0.63	0.68	0.88	0.78	0.75	0.97	1.00	1.00
(8) OCT	0.00	0.17	0.10	0.05	0.18	0.20	0.18	-	0.68	0.15	0.55	0.55	0.49	0.77	0.62	0.65	0.96	0.94	1.00
(9) MPSVMtikhnov	0.01	0.00	0.14	0.06	0.18	0.34	0.43	0.23	-	0.43	0.50	0.57	0.56	0.83	0.83	0.74	0.88	0.98	0.97
(10) OC1	0.00	0.05	0.08	0.01	0.07	0.09	0.40	0.11	0.39	-	0.18	0.53	0.50	0.78	0.61	0.78	0.94	0.97	0.99
(11) QUEST	0.01	0.00	0.08	0.02	0.03	0.05	0.15	0.26	0.24	0.33	-	0.40	0.52	0.60	0.63	0.57	0.75	0.99	0.98
(12) CLM	0.02	0.10	0.04	0.08	0.02	0.15	0.35	0.31	0.30	0.10	0.42	-	0.57	0.63	0.85	0.86	0.94	0.99	0.99
(13) CART	0.00	0.02	0.00	0.00	0.08	0.13	0.29	0.14	0.41	0.27	0.47	0.37	-	0.67	0.84	0.63	0.97	1.00	0.99
(14) CLLVQ	0.01	0.01	0.00	0.00	0.01	0.05	0.07	0.21	0.15	0.12	0.39	0.02	0.21	-	0.15	0.55	0.70	0.96	0.90
(15) CL4	0.00	0.00	0.00	0.00	0.03	0.01	0.06	0.06	0.13	0.01	0.33	0.01	0.07	0.09	-	0.51	0.63	0.92	0.91
(16) LDT	0.00	0.00	0.01	0.01	0.00	0.00	0.09	0.19	0.19	0.09	0.11	0.01	0.31	0.37	0.45	-	0.80	0.96	0.97
(17) CL2	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.12	0.06	0.22	0.01	0.01	0.09	0.22	0.16	-	0.94	0.85
(18) Omni	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.01	-	0.57
(19) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.01	0.02	0.01	0.01	0.10	0.09	0.03	0.15	0.43	-

TABLE 13. Bayesian signed-rank test results for databases with two classes from the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	1.00	0.97	0.98	0.99	0.92	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(2) MPSVMlda	0.00	-	0.57	0.46	0.40	0.57	0.74	0.79	0.86	0.79	0.85	0.97	1.00	1.00	1.00
(3) MPSVMpca	0.03	0.40	-	0.51	0.60	0.73	0.72	0.75	0.96	0.87	0.90	0.99	1.00	1.00	1.00
(4) MPSVMparallel	0.02	0.40	0.48	-	0.51	0.52	0.53	0.90	0.97	0.87	0.87	0.98	1.00	1.00	1.00
(5) QUEST	0.01	0.60	0.40	0.49	-	0.41	0.18	0.75	0.80	0.94	0.97	0.93	0.95	1.00	1.00
(6) OCT	0.04	0.43	0.27	0.47	0.50	-	0.60	0.55	0.84	0.86	0.86	0.80	1.00	1.00	1.00
(7) CRUISE	0.00	0.26	0.28	0.46	0.06	0.36	-	0.56	0.82	0.80	0.88	0.88	0.97	1.00	1.00
(8) MPSVMtikhnov	0.00	0.07	0.24	0.07	0.25	0.45	0.44	-	0.75	0.70	0.75	0.95	0.96	1.00	1.00
(9) CLM	0.00	0.13	0.04	0.03	0.19	0.16	0.17	0.25	-	0.58	0.33	0.70	0.94	1.00	1.00
(10) CLDA	0.00	0.20	0.13	0.13	0.02	0.13	0.05	0.30	0.40	-	0.41	0.62	0.82	0.99	1.00
(11) CLMIX	0.00	0.15	0.10	0.13	0.01	0.14	0.10	0.25	0.32	0.32	-	0.78	0.86	1.00	1.00
(12) MPSVMsubspace	0.00	0.03	0.01	0.02	0.07	0.20	0.12	0.03	0.30	0.38	0.22	-	0.76	0.98	1.00
(13) CLLVQ	0.00	0.00	0.00	0.00	0.05	0.00	0.02	0.04	0.01	0.18	0.12	0.24	-	0.98	1.00
(14) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.02	0.01	-	0.56
(15) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-

TABLE 14. Bayesian signed-rank test results for databases with more than two classes from the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) CRUISE	-	0.39	0.69	0.75	0.84	0.98	0.85	0.97	0.84	1.00	0.58	1.00	1.00	1.00	1.00
(2) MHLDT	0.27	-	0.50	0.26	0.98	0.80	0.58	0.92	0.79	0.96	0.88	0.99	1.00	1.00	1.00
(3) MPSVMlda	0.25	0.07	-	0.14	0.75	0.47	0.50	0.90	0.84	0.98	0.89	0.95	1.00	1.00	1.00
(4) MPSVMparallel	0.20	0.01	0.04	-	0.23	0.57	0.69	0.89	0.46	0.92	0.96	0.97	1.00	1.00	1.00
(5) MPSVMpca	0.15	0.00	0.10	0.03	-	0.40	0.47	0.85	0.17	0.91	0.94	0.95	1.00	1.00	1.00
(6) CLM	0.00	0.04	0.19	0.11	0.24	-	0.45	0.72	0.44	0.73	0.66	0.87	1.00	1.00	1.00
(7) CLMIX	0.01	0.02	0.06	0.26	0.46	0.28	-	0.26	0.65	0.75	0.58	0.79	0.98	0.99	1.00
(8) CLDA	0.01	0.02	0.01	0.07	0.15	0.27	0.04	-	0.66	0.82	0.69	0.86	0.94	1.00	1.00
(9) OCT	0.06	0.00	0.10	0.01	0.09	0.31	0.27	0.32	-	0.74	0.79	0.91	0.96	1.00	1.00
(10) CLLVQ	0.00	0.00	0.01	0.00	0.03	0.00	0.07	0.13	0.22	-	0.56	0.54	0.71	1.00	0.99
(11) QUEST	0.00	0.00	0.00	0.01	0.06	0.25	0.06	0.22	0.14	0.44	-	0.58	0.70	0.82	0.97
(12) MPSVMtikhnov	0.00	0.00	0.00	0.00	0.05	0.10	0.14	0.13	0.09	0.44	0.40	-	0.71	0.89	0.96
(13) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.02	0.30	0.28	-	0.95	0.93
(14) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18	0.10	0.02	-	0.76
(15) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.03	0.04	0.07	0.24	-

the PrefixSpan algorithm and are used to measure similarities between trees. This approach may be extended to MDTs; however, the challenge of comparing multivariate items efficiently must be handled first. When dealing with MDTs with linear combinations, the hyperplane generated in a split might be changed by adding small noise so that the original and new hyperplanes are almost parallel and divide the data in the same way. Hence, we should evaluate when to consider pairs of multivariate items as equivalent even if the weights and splitting points do not match exactly.

5) Training time is a measure of interest not included in this work. We believe that we cannot make a fair comparison of runtime at the moment, given the great diversity of platforms and programming languages used for implementing the algorithms such as Matlab, Weka (Java), C, and Julia. Furthermore, the algorithms implemented in C had to be evaluated in a Linux system; however, due to the volume of the experiments, the rest of the algorithms had to be evaluated in Windows servers of the GIEE-ML group at Tecnologico de Monterrey.

TABLE 15. Bayesian signed-rank test results for databases with up to 20 features from the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.30	0.47	0.29	0.71	0.80	0.90	0.61	0.99	0.91	0.96	0.78	0.97	1.00	1.00	1.00	0.99	1.00	1.00
(2) MPSVMlda	0.25	-	0.68	0.30	0.54	0.60	0.85	0.71	0.97	0.94	0.97	0.96	0.99	1.00	1.00	1.00	0.99	1.00	1.00
(3) CRUISE	0.28	0.15	-	0.43	0.69	0.68	0.59	0.72	0.81	0.76	0.88	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(4) QUEST	0.05	0.06	0.06	-	0.35	0.41	0.53	0.57	0.86	0.72	0.74	0.71	0.92	0.99	1.00	0.99	0.99	1.00	1.00
(5) CLMIX	0.07	0.05	0.06	0.06	-	0.40	0.49	0.56	0.75	0.58	0.75	0.78	0.87	1.00	0.99	0.99	0.97	1.00	1.00
(6) MPSVMtikhnov	0.02	0.00	0.27	0.25	0.40	-	0.43	0.13	0.51	0.72	0.63	0.57	0.91	0.97	0.99	0.98	0.97	1.00	1.00
(7) MPSVMpca	0.05	0.05	0.40	0.44	0.40	0.42	-	0.51	0.51	0.78	0.86	0.87	0.94	0.97	1.00	1.00	0.98	1.00	1.00
(8) MPSVMparallel	0.02	0.00	0.26	0.36	0.41	0.32	0.42	-	0.56	0.88	0.83	0.59	0.88	0.99	0.99	1.00	0.97	1.00	1.00
(9) CART	0.00	0.01	0.09	0.12	0.24	0.45	0.08	0.23	-	0.42	0.56	0.62	0.80	0.97	0.98	0.99	0.84	1.00	1.00
(10) OC1	0.00	0.01	0.03	0.02	0.24	0.18	0.09	0.09	0.22	-	0.62	0.32	0.92	0.93	0.99	0.96	0.92	1.00	1.00
(11) CLDA	0.02	0.02	0.03	0.09	0.16	0.36	0.14	0.16	0.44	0.38	-	0.64	0.82	0.93	0.97	0.96	0.96	1.00	1.00
(12) OCT	0.00	0.03	0.05	0.07	0.15	0.37	0.09	0.05	0.27	0.21	0.35	-	0.76	0.90	0.96	0.97	0.89	1.00	1.00
(13) CLM	0.02	0.00	0.00	0.02	0.00	0.05	0.04	0.08	0.18	0.05	0.18	0.21	-	0.81	0.76	0.91	0.95	1.00	1.00
(14) MPSVMsubspace	0.00	0.00	0.00	0.01	0.00	0.03	0.03	0.01	0.03	0.07	0.07	0.10	0.19	-	0.66	0.65	0.55	0.90	0.85
(15) CLLVQ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.03	0.04	0.01	0.34	-	0.20	0.74	0.95	0.98
(16) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	0.02	0.01	0.35	0.18	-	0.75	0.91	0.98
(17) LDT	0.01	0.01	0.00	0.01	0.01	0.03	0.02	0.02	0.16	0.07	0.04	0.11	0.04	0.45	0.26	0.25	-	0.56	0.94
(18) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.03	0.08	0.44	-	0.96
(19) Omni	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.02	0.02	0.04	0.04	-

TABLE 16. Bayesian signed-rank test results for databases with more than 20 features from the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.66	0.66	0.92	0.98	0.88	0.83	0.95	0.99	0.96	0.97	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(2) OCT	0.14	-	0.45	0.42	0.91	0.68	0.84	0.90	0.83	0.94	0.71	0.99	0.97	0.99	1.00	0.98	1.00	0.99	1.00
(3) OC1	0.22	0.38	-	0.32	0.86	0.54	0.80	0.90	0.91	0.88	0.62	0.90	0.95	1.00	1.00	1.00	1.00	0.99	1.00
(4) CART	0.04	0.15	0.15	-	0.53	0.72	0.68	0.65	0.94	0.87	0.89	0.98	0.99	0.99	1.00	1.00	1.00	0.99	1.00
(5) MPSVMparallel	0.02	0.03	0.02	0.06	-	0.50	0.40	0.15	0.76	0.80	0.63	0.93	0.94	0.99	1.00	0.99	1.00	0.99	1.00
(6) CLM	0.02	0.29	0.16	0.25	0.36	-	0.60	0.47	0.72	0.83	0.67	0.96	0.93	0.98	1.00	1.00	1.00	0.99	1.00
(7) CRUISE	0.00	0.05	0.04	0.23	0.44	0.14	-	0.43	0.62	0.67	0.54	0.82	0.39	0.97	0.98	0.98	0.99	0.98	0.99
(8) MPSVMpca	0.02	0.09	0.09	0.25	0.10	0.42	0.54	-	0.82	0.59	0.69	0.91	0.94	0.98	1.00	1.00	1.00	0.99	1.00
(9) CLMIX	0.01	0.04	0.04	0.05	0.09	0.00	0.17	0.12	-	0.28	0.26	0.74	0.65	0.79	0.95	0.84	0.98	0.98	1.00
(10) CLLVQ	0.00	0.03	0.03	0.12	0.17	0.01	0.27	0.08	0.19	-	0.48	0.58	0.89	0.83	0.97	1.00	0.99	0.98	1.00
(11) CLDA	0.01	0.11	0.08	0.09	0.35	0.01	0.22	0.31	0.23	0.36	-	0.80	0.79	0.88	0.96	0.97	0.96	0.98	1.00
(12) MPSVMlda	0.00	0.01	0.00	0.02	0.02	0.01	0.18	0.08	0.24	0.41	0.13	-	0.66	0.85	0.97	0.98	0.99	0.99	1.00
(13) QUEST	0.00	0.02	0.02	0.01	0.06	0.03	0.04	0.06	0.30	0.10	0.14	0.34	-	0.68	0.57	0.68	0.91	0.96	0.98
(14) MPSVMtikhnov	0.00	0.01	0.00	0.01	0.01	0.02	0.03	0.01	0.20	0.16	0.12	0.10	0.32	-	0.73	0.77	0.81	0.95	0.98
(15) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.03	0.42	0.27	-	0.13	0.81	0.97	0.99
(16) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.02	0.32	0.23	0.02	-	0.73	0.97	0.99
(17) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.02	0.01	0.04	0.01	0.09	0.17	0.18	0.27	-	0.95	0.99
(18) LDT	0.00	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.02	0.02	0.01	0.04	0.05	0.03	0.03	0.05	-	1.00
(19) Omni	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.01	0.01	0.01	0.00	-

TABLE 17. Bayesian signed-rank test results for databases with up to 20 features from the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	0.29	0.40	0.19	0.50	0.91	0.68	0.85	0.96	0.57	0.98	1.00	1.00	1.00	1.00
(2) MPSVMlda	0.14	-	0.57	0.25	0.51	0.74	0.42	0.69	0.94	0.91	0.99	1.00	1.00	1.00	1.00
(3) CRUISE	0.24	0.20	-	0.28	0.75	0.68	0.65	0.84	0.90	0.83	1.00	1.00	1.00	1.00	1.00
(4) QUEST	0.05	0.15	0.05	-	0.66	0.67	0.35	0.55	0.79	0.58	0.96	1.00	1.00	1.00	1.00
(5) MPSVMparallel	0.01	0.00	0.22	0.27	-	0.44	0.21	0.57	0.83	0.49	0.93	0.99	1.00	1.00	1.00
(6) MPSVMpca	0.03	0.03	0.30	0.32	0.44	-	0.33	0.54	0.83	0.77	0.96	0.97	1.00	1.00	1.00
(7) MPSVMtikhnov	0.01	0.00	0.23	0.15	0.10	0.50	-	0.60	0.70	0.47	0.95	0.98	1.00	0.99	1.00
(8) CLMIX	0.03	0.02	0.03	0.04	0.40	0.36	0.25	-	0.62	0.63	0.75	0.99	0.98	0.99	1.00
(9) CLDA	0.01	0.02	0.02	0.05	0.15	0.17	0.29	0.30	-	0.55	0.89	0.95	0.98	0.98	1.00
(10) OCT	0.00	0.06	0.08	0.05	0.08	0.20	0.41	0.30	0.42	-	0.87	0.95	0.98	0.99	1.00
(11) CLM	0.01	0.00	0.00	0.01	0.04	0.03	0.02	0.00	0.11	0.11	-	0.79	0.63	0.90	1.00
(12) MPSVMsubspace	0.00	0.00	0.00	0.00	0.01	0.03	0.02	0.01	0.05	0.05	0.21	-	0.63	0.64	0.88
(13) CLLVQ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.02	0.00	0.37	-	0.24	0.96
(14) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.00	0.36	0.11	-	0.89
(15) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.01	0.07	-

To make a fair comparison of the runtime of MDTs, we propose to implement the MDTs using the same platform and programming language. A statistical comparison should be performed comparing the original implementation with the new implementation of each algorithm to ensure there are no statistically significant differences. Then, it is possible to make a fair comparison of runtime by executing the algorithms on servers with the same characteristics and operating systems.

In Section III-C, we briefly described Model trees, which make multivariate decisions only at the leaves, and Functional trees, which make multivariate decisions at inner nodes and the leaves. It is important to compare the top-ranked algorithms of each group of algorithms, given that they all make multivariate decisions in some of the tree nodes, and can be interchangeably used in some contexts; for example, in contrast pattern-based classification, we could extract multivariate contrast patterns [20] from any of the three types of tree. Gama [52] compared one algorithm of each group without finding statistically significant differences between Functional trees and the other algorithms. However, the algorithms compared by Gama [52] are not shown to be

TABLE 18. Bayesian signed-rank test results for databases with more than 20 features from the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	0.96	0.93	0.99	0.93	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(2) MPSVMparallel	0.03	-	0.51	0.46	0.81	0.68	0.81	0.98	0.93	0.91	0.99	1.00	1.00	1.00	1.00
(3) CRUISE	0.04	0.48	-	0.56	0.63	0.49	0.67	0.93	0.94	0.96	0.32	0.98	1.00	1.00	1.00
(4) MPSVMpca	0.01	0.07	0.44	-	0.61	0.50	0.57	0.89	0.89	0.85	0.98	0.99	1.00	1.00	1.00
(5) CLM	0.04	0.14	0.28	0.35	-	0.68	0.86	0.98	0.93	0.98	0.96	0.99	1.00	1.00	1.00
(6) OCT	0.01	0.19	0.46	0.49	0.21	-	0.57	0.90	0.82	0.79	0.97	0.96	1.00	1.00	1.00
(7) MPSVMlda	0.00	0.10	0.33	0.42	0.12	0.42	-	0.85	0.74	0.72	0.95	0.99	1.00	1.00	1.00
(8) CLLVQ	0.00	0.02	0.07	0.08	0.00	0.08	0.11	-	0.42	0.67	0.93	0.90	1.00	0.98	1.00
(9) CLMIX	0.00	0.07	0.02	0.11	0.00	0.17	0.25	0.32	-	0.04	0.57	0.83	0.99	0.99	1.00
(10) CLDA	0.00	0.09	0.02	0.15	0.01	0.20	0.25	0.26	0.01	-	0.63	0.88	0.99	0.97	1.00
(11) QUEST	0.00	0.01	0.01	0.02	0.03	0.02	0.05	0.07	0.41	0.36	-	0.55	0.62	0.81	0.76
(12) MPSVMtikhnov	0.00	0.00	0.02	0.01	0.01	0.04	0.01	0.10	0.17	0.12	0.45	-	0.81	0.90	0.93
(13) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.38	0.19	-	0.68	0.64
(14) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.03	0.19	0.10	0.32	-	0.69
(15) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.07	0.00	0.31	-

TABLE 19. Bayesian signed-rank test results for databases with up to 1,000 objects from the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.68	0.91	0.97	0.94	0.95	0.96	0.93	0.88	1.00	0.98	0.97	0.99	1.00	1.00	1.00	1.00	1.00	1.00
(2) CRUISE	0.09	-	0.53	0.63	0.68	0.84	0.71	0.77	0.83	0.86	0.86	0.82	0.97	1.00	1.00	1.00	1.00	1.00	1.00
(3) MPSVMlda	0.09	0.44	-	0.52	0.62	0.58	0.63	0.63	0.58	0.63	0.68	0.79	0.91	1.00	1.00	1.00	1.00	1.00	1.00
(4) MPSVMpca	0.02	0.37	0.47	-	0.61	0.62	0.70	0.51	0.71	0.71	0.76	0.84	0.92	0.99	1.00	1.00	1.00	1.00	1.00
(5) CLDA	0.05	0.15	0.35	0.39	-	0.43	0.55	0.40	0.69	0.66	0.55	0.69	0.81	0.94	0.99	0.99	1.00	1.00	1.00
(6) MPSVMparallel	0.01	0.14	0.29	0.35	0.57	-	0.61	0.52	0.42	0.54	0.71	0.77	0.83	0.99	1.00	1.00	1.00	1.00	1.00
(7) OC1	0.03	0.19	0.30	0.28	0.44	0.37	-	0.39	0.49	0.50	0.50	0.52	0.91	0.97	0.99	0.99	1.00	0.99	1.00
(8) CLMIX	0.06	0.12	0.32	0.48	0.32	0.47	0.56	-	0.70	0.65	0.36	0.50	0.75	0.99	0.99	1.00	1.00	1.00	1.00
(9) OCT	0.00	0.15	0.42	0.28	0.30	0.44	0.42	0.26	-	0.49	0.56	0.58	0.75	0.90	0.99	1.00	1.00	0.99	1.00
(10) CART	0.00	0.05	0.37	0.14	0.34	0.34	0.38	0.34	0.36	-	0.70	0.65	0.76	0.98	0.98	0.99	1.00	0.99	1.00
(11) MPSVMtikhnov	0.01	0.14	0.04	0.23	0.45	0.15	0.48	0.61	0.44	0.29	-	0.70	0.83	0.98	1.00	1.00	1.00	1.00	1.00
(12) QUEST	0.02	0.02	0.10	0.16	0.22	0.23	0.42	0.41	0.41	0.35	0.27	-	0.66	0.88	0.87	0.92	0.97	0.99	1.00
(13) CLM	0.01	0.03	0.09	0.07	0.19	0.16	0.08	0.09	0.25	0.24	0.12	0.32	-	0.91	0.96	1.00	1.00	0.99	1.00
(14) MPSVMsubspace	0.00	0.00	0.00	0.01	0.06	0.01	0.03	0.01	0.10	0.02	0.01	0.12	0.09	-	0.72	0.78	0.93	0.95	0.99
(15) CLLVQ	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.01	0.02	0.00	0.13	0.01	0.28	-	0.44	0.97	0.96	1.00
(16) CL4	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.08	0.00	0.22	0.08	-	0.80	0.96	1.00
(17) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.07	0.02	0.17	-	0.90	1.00
(18) LDT	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.01	0.01	0.05	0.04	0.04	0.10	-	0.99
(19) Omni	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	-

TABLE 20. Bayesian signed-rank test results for databases with more than 1,000 objects from the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.72	0.39	0.52	0.74	0.75	0.29	0.57	0.63	0.89	0.78	0.85	0.97	0.99	0.98	1.00	0.88	1.00	0.96
(2) CART	0.07	-	0.47	0.11	0.14	0.48	0.61	0.79	0.75	0.49	0.86	0.82	0.99	0.95	0.99	0.99	0.89	1.00	0.98
(3) OCT	0.05	0.19	-	0.16	0.26	0.66	0.26	0.44	0.60	0.20	0.60	0.77	0.66	0.94	0.91	0.96	0.83	1.00	0.88
(4) OC1	0.04	0.01	0.07	-	0.30	0.46	0.21	0.35	0.49	0.39	0.69	0.71	0.85	0.93	0.98	0.99	0.86	1.00	0.87
(5) MPSVMparallel	0.05	0.04	0.20	0.19	-	0.29	0.39	0.64	0.40	0.10	0.44	0.62	0.88	0.86	0.96	0.99	0.87	1.00	0.96
(6) MPSVMpca	0.04	0.10	0.20	0.15	0.05	-	0.42	0.57	0.46	0.20	0.50	0.43	0.95	0.67	0.97	0.98	0.81	1.00	0.96
(7) QUEST	0.00	0.31	0.05	0.28	0.25	0.28	-	0.10	0.23	0.50	0.31	0.52	0.83	0.87	0.97	0.97	0.82	1.00	0.92
(8) CRUISE	0.01	0.15	0.11	0.11	0.18	0.35	0.02	-	0.38	0.58	0.56	0.84	0.87	0.88	0.95	0.96	0.90	0.96	0.94
(9) CLM	0.02	0.15	0.12	0.08	0.12	0.26	0.31	0.15	-	0.55	0.21	0.30	0.75	0.90	0.95	0.97	0.92	0.99	0.94
(10) MPSVMlda	0.01	0.41	0.16	0.27	0.28	0.48	0.21	0.37	0.21	-	0.21	0.34	0.80	0.81	0.96	0.94	0.81	1.00	0.96
(11) CLMIX	0.00	0.10	0.04	0.05	0.10	0.10	0.03	0.05	0.05	0.28	-	0.12	0.82	0.89	0.93	0.89	0.59	0.99	0.93
(12) CLLVQ	0.01	0.06	0.06	0.03	0.10	0.12	0.15	0.04	0.00	0.26	0.05	-	0.60	0.65	0.91	0.96	0.88	0.99	0.83
(13) CLDA	0.00	0.00	0.02	0.00	0.02	0.04	0.02	0.04	0.03	0.03	0.05	0.32	-	0.72	0.85	0.96	0.61	0.96	0.90
(14) MPSVMtikhnov	0.00	0.03	0.01	0.03	0.01	0.03	0.02	0.08	0.07	0.01	0.06	0.11	0.28	-	0.60	0.62	0.62	0.64	0.87
(15) CL4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.01	0.05	0.35	-	0.41	0.55	0.94	0.80
(16) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.33	0.00	-	0.36	0.92	0.71
(17) LDT	0.01	0.07	0.06	0.05	0.04	0.01	0.08	0.06	0.02	0.06	0.02	0.03	0.33	0.36	0.13	0.16	-	0.62	0.60
(18) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.01	0.00	0.01	0.01	0.04	0.36	0.06	0.07	0.38	-	0.63
(19) Omni	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.03	0.12	0.01	0.12	0.10	0.37	-

among the top-ranked in their respective groups, and more MDT and Model tree algorithms have been published since Gama's comparison. Therefore, Gama's comparison should be updated to include the top-ranked algorithms of each group.

In this work, we only compared single MDTs. However, authors have built ensembles using MDTs, such as random forests (Oblique Random Forest [43]), multivariate alternating decision trees [63], and pattern-based classifiers (Pattern-based classifier for class imbalance problems PBC4cip with MHLDT [20]). Further work is thus concerned with comparing different MDT-based ensembles to identify which strategies lead to better classification performance.

APPENDIX A DATABASES

Here we list each database used in the experimental comparison of algorithms in Table 9. All databases, except dataset3d, come from the UCI repository [12]. References for the databases with a citation request are included in the table. For each database, we show the number of classes, number of features, number of objects, and degree of

TABLE 21. Bayesian signed-rank test results for databases with up to 1,000 objects from the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	0.95	0.93	0.79	0.93	0.99	0.91	0.98	0.99	0.98	0.99	1.00	1.00	1.00	1.00
(2) MPSVMpca	0.04	-	0.54	0.54	0.50	0.84	0.84	0.78	0.86	0.94	0.95	1.00	1.00	1.00	1.00
(3) MPSVMlda	0.07	0.46	-	0.62	0.46	0.85	0.64	0.85	0.90	0.88	0.94	1.00	1.00	1.00	1.00
(4) CRUISE	0.06	0.46	0.37	-	0.59	0.79	0.78	0.83	0.84	0.93	0.72	0.99	1.00	1.00	1.00
(5) MPSVMparallel	0.03	0.48	0.43	0.40	-	0.86	0.72	0.83	0.89	0.93	0.95	1.00	1.00	1.00	1.00
(6) CLDA	0.01	0.16	0.14	0.08	0.14	-	0.51	0.24	0.51	0.62	0.67	0.94	0.95	0.98	1.00
(7) OCT	0.00	0.16	0.36	0.18	0.20	0.48	-	0.41	0.56	0.63	0.71	0.98	0.95	1.00	1.00
(8) CLMIX	0.02	0.22	0.12	0.06	0.17	0.19	0.52	-	0.38	0.54	0.52	0.95	0.99	0.99	1.00
(9) MPSVMtikhnov	0.01	0.14	0.03	0.16	0.04	0.49	0.43	0.61	-	0.73	0.74	0.97	0.99	1.00	1.00
(10) CLM	0.01	0.05	0.11	0.05	0.07	0.38	0.35	0.23	0.24	-	0.50	0.97	0.98	1.00	1.00
(11) QUEST	0.01	0.05	0.03	0.00	0.05	0.23	0.29	0.36	0.25	0.48	-	0.65	0.91	0.83	0.93
(12) CLLVQ	0.00	0.00	0.00	0.01	0.00	0.06	0.02	0.04	0.03	0.01	0.35	-	0.57	0.64	0.99
(13) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.05	0.05	0.01	0.00	0.02	0.09	0.43	-	0.58	0.86
(14) CL4	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.17	0.06	0.42	-	0.90
(15) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.01	0.14	0.07	-

TABLE 22. Bayesian signed-rank test results for databases with more than 1,000 objects from the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	0.69	0.71	0.93	0.75	0.71	0.99	0.96	0.98	1.00	0.99	1.00	1.00	1.00	1.00
(2) CRUISE	0.16	-	0.64	0.75	0.05	0.59	0.78	0.96	0.94	0.98	0.99	0.98	1.00	1.00	1.00
(3) MPSVMparallel	0.01	0.30	-	0.02	0.63	0.44	0.31	0.74	0.78	0.89	0.97	0.97	1.00	1.00	1.00
(4) MPSVMlda	0.00	0.24	0.17	-	0.42	0.73	0.69	0.53	0.66	0.86	0.78	0.90	1.00	1.00	1.00
(5) QUEST	0.00	0.06	0.29	0.56	-	0.14	0.44	0.53	0.74	0.95	0.81	0.93	0.99	0.99	0.99
(6) OCT	0.00	0.29	0.05	0.10	0.51	-	0.56	0.51	0.73	0.75	0.88	0.94	1.00	1.00	1.00
(7) MPSVMpca	0.00	0.21	0.03	0.02	0.54	0.25	-	0.65	0.70	0.82	0.61	0.88	1.00	1.00	1.00
(8) CLM	0.02	0.02	0.08	0.27	0.38	0.28	0.23	-	0.47	0.89	0.53	0.88	1.00	0.99	1.00
(9) CLMIX	0.00	0.01	0.20	0.15	0.06	0.25	0.27	0.08	-	0.47	0.31	0.81	1.00	0.98	0.99
(10) CLDA	0.00	0.01	0.08	0.03	0.03	0.22	0.18	0.10	0.09	-	0.56	0.75	0.99	0.97	1.00
(11) CLLVQ	0.00	0.00	0.01	0.04	0.14	0.05	0.09	0.00	0.11	0.39	-	0.65	0.99	0.96	1.00
(12) MPSVMtikhnov	0.00	0.02	0.01	0.00	0.07	0.06	0.08	0.12	0.18	0.24	0.28	-	0.84	0.76	0.94
(13) CL4	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.16	-	0.58	0.62
(14) MPSVMsubspace	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.02	0.03	0.04	0.24	0.42	-	0.73
(15) CL2	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.27	-

TABLE 23. Bayesian signed-rank test results for databases with up to two objects of the majority class for each object from the minority class, for the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.30	0.62	0.50	0.46	0.58	0.33	0.54	0.84	0.88	0.51	0.77	0.88	0.97	0.99	1.00	1.00	1.00	1.00
(2) CRUISE	0.03	-	0.14	0.53	0.55	0.39	0.47	0.81	0.73	0.76	0.71	0.12	0.94	0.99	0.99	1.00	1.00	1.00	1.00
(3) CLMIX	0.04	0.09	-	0.25	0.20	0.13	0.40	0.44	0.47	0.56	0.49	0.21	0.75	0.85	0.98	0.93	1.00	1.00	1.00
(4) CLDA	0.03	0.19	0.18	-	0.46	0.41	0.45	0.57	0.56	0.45	0.63	0.39	0.81	0.96	0.97	0.99	1.00	1.00	1.00
(5) OCT	0.03	0.30	0.11	0.36	-	0.47	0.14	0.19	0.23	0.71	0.74	0.63	0.96	0.94	0.94	0.99	1.00	1.00	1.00
(6) CLM	0.03	0.19	0.01	0.41	0.32	-	0.23	0.35	0.46	0.56	0.30	0.29	0.55	0.79	0.97	0.99	1.00	1.00	1.00
(7) OC1	0.02	0.14	0.07	0.35	0.02	0.13	-	0.47	0.16	0.65	0.58	0.37	0.75	0.91	0.77	1.00	1.00	1.00	1.00
(8) MPSVMparallel	0.01	0.15	0.12	0.36	0.00	0.23	0.15	-	0.28	0.52	0.13	0.51	0.75	0.86	0.97	0.98	1.00	1.00	1.00
(9) CART	0.02	0.08	0.21	0.43	0.02	0.31	0.04	0.20	-	0.63	0.26	0.61	0.83	0.90	0.93	0.99	1.00	1.00	1.00
(10) MPSVMlda	0.01	0.08	0.00	0.04	0.08	0.09	0.06	0.22	0.35	-	0.45	0.18	0.70	0.82	0.97	0.99	1.00	1.00	1.00
(11) MPSVMpca	0.02	0.26	0.13	0.37	0.04	0.23	0.11	0.03	0.14	0.45	-	0.62	0.38	0.69	0.97	0.99	1.00	1.00	1.00
(12) QUEST	0.00	0.00	0.04	0.36	0.21	0.16	0.18	0.44	0.31	0.32	0.35	-	0.76	0.80	0.92	0.96	1.00	1.00	1.00
(13) CLLVQ	0.00	0.04	0.01	0.17	0.02	0.00	0.02	0.05	0.14	0.18	0.01	0.20	-	0.47	0.71	0.96	0.99	1.00	1.00
(14) MPSVMtikhnov	0.00	0.00	0.00	0.04	0.02	0.06	0.04	0.02	0.08	0.03	0.04	0.05	0.23	-	0.88	0.93	0.95	1.00	1.00
(15) CL4	0.00	0.01	0.00	0.02	0.00	0.00	0.00	0.00	0.02	0.01	0.00	0.05	0.00	0.11	-	0.22	0.88	1.00	0.99
(16) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.03	0.00	0.05	0.06	-	0.85	1.00	1.00
(17) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.11	0.15	-	0.96	0.98
(18) LDT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	-	0.91
(19) Omni	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.04	-

class imbalance. The subset of databases used in the full algorithms experiment (see Section IV-D) can be distinguished in the table.

The names of the databases match exactly the names in the UCI repository; however, the number of objects and features may be different. The number of objects may differ because some databases have separate training and testing sets and only the number of objects of one set is reported, or because of inconsistencies between the reported number of objects and the actual number of objects in the files. The number of features may differ because some authors count the class as an additional feature (we do not), or some features, such as IDs, must be removed.

APPENDIX B
CLASSIFIER PERFORMANCE AND FULL BAYESIAN SIGNED-RANK TEST RESULTS

In Section V, we showed the distribution of AUC for each classifier. In Table 10, we show the AUC of each classifier in the 57 databases. The subset of databases used in the full algorithms experiment (see Section IV-D) can be distinguished in the table.

TABLE 24. Bayesian signed-rank test results for databases with more than two objects of the majority class for each object from the minority class, for the full algorithms experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
(1) MHLDT	-	0.85	0.97	1.00	0.98	0.97	0.92	0.99	0.90	0.93	0.98	1.00	0.99	0.90	0.99	1.00	1.00	1.00	0.98
(2) MPSVMlda	0.14	-	0.68	0.76	0.67	0.74	0.83	0.90	0.91	0.93	0.99	1.00	0.99	0.89	1.00	1.00	1.00	1.00	0.97
(3) MPSVMpca	0.03	0.28	-	0.78	0.70	0.69	0.68	0.86	0.82	0.93	0.92	0.98	0.99	0.89	0.98	1.00	1.00	1.00	0.97
(4) CART	0.00	0.21	0.18	-	0.48	0.79	0.52	0.66	0.62	0.85	0.78	0.90	0.93	0.64	0.91	0.99	0.99	1.00	0.97
(5) MPSVMparallel	0.02	0.02	0.30	0.27	-	0.69	0.61	0.78	0.82	0.80	0.91	0.96	0.96	0.83	0.99	1.00	1.00	1.00	0.97
(6) MPSVMtikhnov	0.03	0.00	0.31	0.21	0.18	-	0.55	0.56	0.77	0.54	0.89	0.90	0.92	0.77	0.92	0.98	0.99	0.99	0.94
(7) CRUISE	0.06	0.17	0.32	0.46	0.32	0.44	-	0.44	0.66	0.67	0.95	0.93	0.99	0.88	0.95	1.00	1.00	1.00	0.96
(8) OC1	0.01	0.06	0.14	0.29	0.22	0.43	0.46	-	0.37	0.68	0.86	0.88	0.97	0.67	0.90	0.97	0.99	0.99	0.95
(9) QUEST	0.04	0.07	0.18	0.38	0.17	0.22	0.20	0.60	-	0.49	0.76	0.70	0.79	0.69	0.83	0.89	0.97	0.99	0.95
(10) OCT	0.00	0.07	0.07	0.14	0.19	0.46	0.29	0.31	0.42	-	0.52	0.62	0.82	0.56	0.73	0.95	0.98	1.00	0.96
(11) CLMIX	0.02	0.01	0.08	0.22	0.09	0.11	0.05	0.13	0.23	0.48	-	0.81	0.62	0.69	0.81	0.92	0.97	1.00	0.94
(12) CLDA	0.00	0.00	0.02	0.10	0.04	0.10	0.02	0.12	0.24	0.38	0.18	-	0.55	0.61	0.65	0.94	0.96	1.00	0.96
(13) CLM	0.01	0.01	0.01	0.07	0.04	0.08	0.01	0.03	0.21	0.18	0.37	0.44	-	0.71	0.54	0.96	0.99	1.00	0.93
(14) LDT	0.10	0.11	0.11	0.36	0.17	0.23	0.12	0.33	0.31	0.44	0.30	0.39	0.29	-	0.54	0.60	0.67	0.87	0.99
(15) MPSVMsubspace	0.01	0.00	0.02	0.09	0.01	0.08	0.05	0.10	0.17	0.27	0.19	0.35	0.46	0.46	-	0.82	0.81	0.96	0.88
(16) CLLVQ	0.00	0.00	0.00	0.01	0.00	0.02	0.00	0.03	0.11	0.05	0.08	0.05	0.02	0.40	0.18	-	0.44	0.97	0.89
(17) CL4	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.03	0.02	0.03	0.03	0.01	0.33	0.19	0.37	-	0.95	0.90
(18) CL2	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.13	0.04	0.03	0.05	-	0.88
(19) Omni	0.02	0.03	0.03	0.03	0.03	0.06	0.04	0.05	0.05	0.04	0.06	0.04	0.07	0.01	0.12	0.11	0.10	0.12	-

TABLE 25. Bayesian signed-rank test results for databases with up to two objects of the majority class for each object from the minority class, for the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	0.36	0.73	0.64	0.72	0.65	0.58	0.89	0.77	0.71	0.94	0.95	1.00	1.00	1.00
(2) CRUISE	0.06	-	0.05	0.49	0.18	0.78	0.92	0.78	0.84	0.79	0.98	0.99	1.00	1.00	1.00
(3) QUEST	0.01	0.01	-	0.54	0.13	0.54	0.77	0.69	0.71	0.65	0.91	0.95	0.99	1.00	1.00
(4) CLDA	0.03	0.21	0.21	-	0.14	0.69	0.72	0.34	0.81	0.75	0.95	0.94	1.00	1.00	1.00
(5) CLMIX	0.02	0.11	0.10	0.12	-	0.54	0.71	0.43	0.74	0.30	0.80	0.87	1.00	0.99	1.00
(6) OCT	0.01	0.17	0.35	0.25	0.13	-	0.14	0.59	0.68	0.49	0.84	0.96	0.96	1.00	1.00
(7) MPSVMparallel	0.00	0.06	0.22	0.20	0.10	0.02	-	0.29	0.10	0.36	0.67	0.80	1.00	1.00	1.00
(8) MPSVMlda	0.00	0.10	0.08	0.02	0.01	0.29	0.28	-	0.65	0.26	0.57	0.85	1.00	1.00	1.00
(9) MPSVMpca	0.00	0.13	0.28	0.18	0.09	0.08	0.05	0.23	-	0.39	0.55	0.37	1.00	1.00	1.00
(10) CLM	0.01	0.06	0.13	0.17	0.00	0.27	0.18	0.32	0.23	-	0.56	0.46	0.99	1.00	1.00
(11) MPSVMtikhnov	0.00	0.01	0.02	0.04	0.01	0.11	0.07	0.02	0.20	0.22	-	0.49	0.98	0.99	0.97
(12) CLLVQ	0.00	0.01	0.05	0.05	0.00	0.01	0.01	0.06	0.00	0.00	0.24	-	0.89	0.99	0.95
(13) CL4	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	-	0.24	0.64
(14) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	-	0.54
(15) MPSVMsubspace	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.05	0.36	0.46	-

TABLE 26. Bayesian signed-rank test results for databases with more than two objects of the majority class for each object from the minority class, for the full databases experiment. Each cell shows the probability that the algorithm in the row is practically better than the algorithm in the column. For a pair of algorithms i, j, the probability of a tie is $1 - p_{ij} - p_{ji}$. The algorithms are ranked by the number of times their probability of winning against another algorithm is higher than their probability of losing.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) MHLDT	-	0.90	0.93	0.98	0.87	0.92	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(2) MPSVMlda	0.10	-	0.48	0.78	0.72	0.97	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(3) MPSVMparallel	0.06	0.13	-	0.69	0.75	0.96	0.98	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00
(4) MPSVMpca	0.02	0.18	0.30	-	0.61	0.92	0.97	0.96	0.99	0.98	0.99	1.00	1.00	1.00	1.00
(5) CRUISE	0.12	0.28	0.24	0.39	-	0.69	0.99	0.84	0.68	1.00	1.00	1.00	0.99	1.00	1.00
(6) OCT	0.00	0.03	0.02	0.08	0.26	-	0.62	0.72	0.89	0.77	0.85	0.93	0.95	0.99	1.00
(7) CLM	0.01	0.04	0.02	0.03	0.01	0.37	-	0.53	0.69	0.89	0.93	0.99	0.92	1.00	1.00
(8) MPSVMtikhnov	0.00	0.00	0.00	0.04	0.16	0.28	0.47	-	0.81	0.76	0.75	0.83	0.93	0.97	0.99
(9) QUEST	0.00	0.00	0.00	0.01	0.02	0.08	0.31	0.19	-	0.69	0.68	0.42	0.63	0.81	0.92
(10) CLMIX	0.00	0.00	0.01	0.02	0.00	0.23	0.10	0.24	0.31	-	0.57	0.58	0.85	0.92	1.00
(11) CLDA	0.00	0.00	0.00	0.01	0.00	0.15	0.07	0.25	0.31	0.20	-	0.63	0.76	0.92	1.00
(12) CLLVQ	0.00	0.00	0.00	0.00	0.00	0.07	0.01	0.17	0.58	0.41	0.37	-	0.58	0.83	1.00
(13) MPSVMsubspace	0.00	0.00	0.00	0.00	0.01	0.05	0.08	0.07	0.37	0.15	0.24	0.42	-	0.61	0.96
(14) CL4	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	0.19	0.07	0.07	0.12	0.39	-	0.99
(15) CL2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.08	0.00	0.00	0.00	0.04	0.00	-

In Section V-A, we presented the full results of the Bayesian signed-rank test for all databases with Tables 7 and 8. The tables show the probability that each algorithm wins and loses against the rest. In Section V-B, we showed plots summarizing the Bayesian signed-rank test in subsets of databases according to the number of classes, number of features, number of objects, and degree of class imbalance. Here, we show the full results of the Bayesian signed-rank test in table format. The results according to the number of classes are shown in Tables 11 - 14. The results according to the number of features are shown in Tables 15 - 18. The results according to the number of objects are shown in Tables 19 - 22. The results according to the degree of imbalance are shown in Tables 23 - 26.

ACKNOWLEDGMENT

The authors would like to thank Eibe Frank for reading our article and his valuable suggestions, specifically including related decision tree approaches and clarifying the correct category for Logistic Model Trees. They would also wish to express their gratitude to the members of the Grupo de Investigación con Enfoque Estratégico-Machine Learning (GIEE-ML) Group, Tecnologico de Monterrey, for providing computational resources for the experiments, useful suggestions and advice on earlier versions of the results presented in this article.

REFERENCES

[1] O. Loyola-Gonzalez, "Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7, pp. 154096–154113, 2019, doi: 10.1109/ACCESS.2019.2949286.

[2] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, "An up-to-date comparison of state-of-the-art classification algorithms," *Expert Syst. Appl.*, vol. 82, pp. 128–150, Oct. 2017, doi: 10.1016/j.eswa.2017.04.003.

[3] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[5] K. Coussement, M. Phan, A. De Caigny, D. F. Benoit, and A. Raes, "Predicting Student dropout in subscription-based online learning environments: The beneficial impact of the logit leaf model," *Decis. Support Syst.*, vol. 135, Aug. 2020, Art. no. 113325. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923620300804

[6] H. Xia, X. Pan, Y. Zhou, and Z. J. Zhang, "Creating the best first impression: Designing online product photos to increase sales," *Decis. Support Syst.*, vol. 131, Apr. 2020, Art. no. 113235. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923619302647

[7] X. Mao, X. Zhao, and Y. Liu, "MHealth app recommendation based on the prediction of suitable behavior change techniques," *Decis. Support Syst.*, vol. 132, May 2020, Art. no. 113248. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923620300038

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. 2nd ed. Hoboken, NJ, USA: Wiley, 2000.

[9] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artif. Intell. Res.*, vol. 2, pp. 1–32, Aug. 1994, doi: 10.1613/jair.63.

[10] X.-B. Li, "A scalable decision tree system and its application in pattern recognition and intrusion detection," *Decis. Support Syst.*, vol. 41, no. 1, pp. 112–130, Nov. 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923604001265

[11] O. T. Yildiz and E. Alpaydin, "Linear discriminant trees," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 3, pp. 323–353, May 2005, doi: 10.1142/S0218001405004125.

[12] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml

[13] G. Santafe, I. Inza, and J. A. Lozano, "Dealing with the evaluation of supervised classification algorithms," *Artif. Intell. Rev.*, vol. 44, no. 4, pp. 467–508, 2015, doi: 10.1007/s10462-015-9433-y.

[14] D. Bertsimas and J. Dunn, "Optimal classification trees," *Mach. Learn.*, vol. 106, no. 7, pp. 1039–1082, Jul. 2017, doi: 10.1007/s10994-017-5633-9.

[15] C. E. Brodley and P. E. Utgoff, "Multivariate decision trees," *Mach. Learn.*, vol. 19, no. 1, pp. 45–77, 1995, doi: 10.1023/A:1022607123649.

[16] V. A. S. Hernández, R. Monroy, M. A. Medina-Pérez, O. Loyola-González, and F. Herrera, "A practical tutorial for decision tree induction: Evaluation measures for candidate splits and opportunities," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–38, Apr. 2021, doi: 10.1145/3429739.

[17] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica Sinica*, vol. 7, no. 4, pp. 815–840, Oct. 1997.

[18] H. Kim and W.-Y. Loh, "Classification trees with unbiased multiway splits," *J. Amer. Stat. Assoc.*, vol. 96, no. 454, pp. 589–604, Jun. 2001.

[19] M. F. Amasyali and O. Ersoy, "Cline: A new decision-tree family," *IEEE Trans. Neural Netw.*, vol. 19, no. 2, pp. 356–363, Feb. 2008, doi: 10.1109/TNN.2007.910729.

[20] L. Canete-Sifuentes, R. Monroy, M. A. Medina-Perez, O. Loyola-Gonzalez, and F. Vera Voronisky, "Classification based on multivariate contrast patterns," *IEEE Access*, vol. 7, pp. 55744–55762, 2019.

[21] Friedman, "A recursive partitioning decision rule for nonparametric classification," *IEEE Trans. Comput.*, vol. C-26, no. 4, pp. 404–408, Apr. 1977.

[22] J. Gama, "Probabilistic linear tree," in *Proc. 14th Int. Conf. Mach. Learn. (ICML)*, Nashville, TN, USA, Jul. 1997, pp. 134–142.

[23] X.-B. Li, J. R. Sweigart, J. T. C. Teng, J. M. Donohue, L. A. Thombs, and S. M. Wang, "Multivariate decision trees using linear discriminants and Tabu search," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 33, no. 2, pp. 194–205, Mar. 2003, doi: 10.1109/TSMCA.2002.806499.

[24] W.-Y. Loh, "Improving the precision of classification trees," *Ann. Appl. Statist.*, vol. 3, no. 4, pp. 1710–1737, Dec. 2009.

[25] N. Manwani and P. S. Sastry, "Geometric decision tree," *IEEE Trans. Syst., Man, Cybern, B, Cybern.*, vol. 42, no. 1, pp. 181–192, Feb. 2012, doi: 10.1109/TSMCB.2011.2163392.

[26] A. López-Chau, J. Cervantes, L. López-García, and F. García-Lamont, "Fisher's decision tree," *Expert Syst. Appl.*, vol. 40, no. 16, pp. 6283–6291, 2013, doi: 10.1016/j.eswa.2013.05.044.

[27] D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. Brown, "HHCART: An oblique decision tree," *Comput. Statist. Data Anal.*, vol. 96, pp. 12–23, Aug. 2016, doi: 10.1016/j.csda.2015.11.006.

[28] F. Wang, Q. Wang, F. Nie, W. Yu, and R. Wang, "Efficient tree classifiers for large scale datasets," *Neurocomputing*, vol. 284, pp. 70–79, Apr. 2018, doi: 10.1016/j.neucom.2017.12.061.

[29] H. Kim and W.-Y. Loh, "Classification trees with bivariate linear discriminant node models," *J. Comput. Graph. Statist.*, vol. 12, no. 3, pp. 512–530, Sep. 2003.

[30] W.-Y. Loh, "Fifty years of classification and regression trees," *Int. Stat. Rev.*, vol. 82, no. 3, pp. 329–348, Dec. 2014.

[31] J. Gama, "Discriminant trees," in *Proc. 16th Int. Conf. Mach. Learn. (ICML)*, Bled, Slovenia, Jun. 1999, pp. 134–142.

[32] L. Zhang and P. N. Suganthan, "Oblique decision tree ensemble via multisurface proximal support vector machine," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2165–2176, Oct. 2015, doi: 10.1109/TCYB.2014.2366468.

[33] D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. A. Brown, "A reflected feature space for CART," *Austral. New Zealand J. Statist.*, vol. 61, no. 3, pp. 380–391, Sep. 2019.

[34] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification Regression Trees*. Belmont, CA, USA: Wadsworth, 1984.

[35] C. T. Yildiz and E. Alpaydin, "Omnivariate decision trees," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1539–1546, Nov. 2001, doi: 10.1109/72.963795.

[36] H. Guo and S. B. Gelfand, "Classification trees with neural network feature extraction," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 923–933, Nov. 1992, doi: 10.1109/72.165594.

[37] D. G. Heath, S. Kasif, and S. Salzberg, "Induction of oblique decision trees," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Chambéry, France, Aug. 1993, pp. 1002–1007.

[38] H. Liu and R. Setiono, "Feature transformation and multivariate decision tree induction," in *Proc. Discovery Sci., 1st Int. Conf. (DS)*, Fukuoka, Japan, Dec. 1998, pp. 279–290, doi: 10.1007/3-540-49292-5_25.

[39] S. Shah and P. S. Sastry, "New algorithms for learning and pruning oblique decision trees," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 29, no. 4, pp. 494–505, Nov. 1999, doi: 10.1109/5326.798764.

[40] L. Bobrowski and M. Kretowski, "Induction of multivariate decision trees by using dipolar criteria," in *Proc. Princ. Data Mining Knowl. Discovery, 4th Eur. Conf., (PKDD)*, Lyon, France, Sep. 2000, pp. 331–336, doi: 10.1007/3-540-45372-5_33.

[41] K. P. Bennett, N. Cristianini, J. Shawe-Taylor, and D. Wu, "Enlarging the margins in perceptron decision trees," *Mach. Learn.*, vol. 41, no. 3, pp. 295–313, 2000, doi: 10.1023/A:1007600130808.

[42] M. Better, F. Glover, and M. Samorani, "Classification by vertical and cutting multi-hyperplane decision tree induction," *Decis. Support Syst.*, vol. 48, no. 3, pp. 430–436, Feb. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923609001407

[43] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Köthe, and F. A. Hamprecht, "On oblique random forests," in *Proc. Mach. Learn. Knowl. Discovery Databases (PKDD)*, Athens, Greece, Sep. 2011, pp. 453–469, doi: 10.1007/978-3-642-23783-6_29.

[44] R. J. R. Struharik, V. Vranjkovic, S. Dautovic, and L. A. Novak, ''Inducing oblique decision trees,'' in *Proc. IEEE 12th Int. Symp. Intell. Syst. Inform. (SISY)*, Subotica, Serbia, Sep. 2014, pp. 257–262, doi: 10.1109/SISY.2014.6923596.

[45] W. Liu and I. W. Tsang, ''Making decision trees feasible in ultrahigh feature and label dimensions,'' *J. Mach. Learn. Res.*, vol. 18, pp. 81:1–81:36, Jan. 2017. [Online]. Available: https://jmlr.org/papers/v18/16-466.html

[46] A. Magana-Mora and V. B. Bajic, ''OmniGA: Optimized omnivariate decision trees for generalizable classification models,'' *Sci. Rep.*, vol. 7, no. 1, p. 3898, Dec. 2017.

[47] F. Nie, W. Zhu, and X. Li, ''Decision tree SVM: An extension of linear SVM for non-linear classification,'' *Neurocomputing*, vol. 401, pp. 153–159, Aug. 2020, doi: 10.1016/j.neucom.2019.10.051.

[48] F. Wang, Q. Wang, F. Nie, Z. Li, W. Yu, and F. Ren, ''A linear multivariate binary decision tree classifier based on K-means splitting,'' *Pattern Recognit.*, vol. 107, Nov. 2020, Art. no. 107521, doi: 10.1016/j.patcog.2020.107521.

[49] O. Irsoy, O. T. Yildiz, and E. Alpaydin, ''Soft decision trees,'' in *Proc. 21st Int. Conf. Pattern Recognit. (ICPR)*, Tsukuba, Japan, Nov. 2012, pp. 1819–1822. [Online]. Available: https://ieeexplore.ieee.org/document/6460506/

[50] A. Suárez and J. F. Lutsko, ''Globally optimal fuzzy decision trees for classification and regression,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 12, pp. 1297–1311, Dec. 1999, doi: 10.1109/34.817409.

[51] N. Landwehr, M. Hall, and E. Frank, ''Logistic model trees,'' *Mach. Learn.*, vol. 59, no. 1, pp. 161–205, May 2005, doi: 10.1007/s10994-005-0466-3.

[52] J. Gama, ''Functional trees,'' *Mach. Learn.*, vol. 55, no. 3, pp. 219–250, Jun. 2004, doi: 10.1023/B:MACH.0000027782.67192.13.

[53] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*. San Mateo, CA, USA: Morgan Kaufmann, 1999. [Online]. Available: https://www.cs.waikato.ac.nz/%7Eml/weka/book.html

[54] J. Derrac, S. García, D. Molina, and F. Herrera, ''A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,'' *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011, doi: 10.1016/j.swevo.2011.02.002.

[55] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[56] A. Benavoli, G. Corani, J. Demsar, and M. Zaffalon, ''Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis,'' *J. Mach. Learn. Res.*, vol. 18, pp. 77:1–77:36, Jan. 2017. [Online]. Available: http://jmlr.org/papers/v18/16-305.html

[57] J. G. Moreno-Torres, J. A. Sáez, and F. Herrera, ''Study on the impact of partition-induced dataset shift on K-fold cross-validation,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1304–1312, Jun. 2012, doi: 10.1109/TNNLS.2012.2199516.

[58] V. López, A. Fernández, and F. Herrera, ''On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed,'' *Inf. Sci.*, vol. 257, pp. 1–13, Feb. 2014, doi: 10.1016/j.ins.2013.09.038.

[59] A. E. Gutierrez-Rodríguez, J. F. Martínez-Trinidad, M. García-Borroto, and J. A. Carrasco-Ochoa, ''Mining patterns for clustering on numerical datasets using unsupervised decision trees,'' *Knowl.-Based Syst.*, vol. 82, pp. 70–79, Jul. 2015, doi: 10.1016/j.knosys.2015.02.019.

[60] O. Loyola-Gonzalez, A. E. Gutierrez-Rodriguez, M. A. Medina-Perez, R. Monroy, J. F. Martinez-Trinidad, J. A. Carrasco-Ochoa, and M. Garcia-Borroto, ''An explainable artificial intelligence model for clustering numerical databases,'' *IEEE Access*, vol. 8, pp. 52370–52384, 2020, doi: 10.1109/ACCESS.2020.2980581.

[61] J. Tanha, M. van Someren, and H. Afsarmanesh, ''Semi-supervised self-training for decision tree classifiers,'' *Int. J. Mach. Learn. Cybern.*, vol. 8, no. 1, pp. 355–370, Feb. 2015, doi: 10.1007/s13042-015-0328-7.

[62] G. Bakirli and D. Birant, ''DTreeSim: A new approach to compute decision tree similarity using re-mining,'' *Turkish J. Electr. Eng. Comput. Sci.*, vol. 25, pp. 108–125, Jan. 2017, doi: 10.3906/elk-1504-234.

[63] H. K. Sok, M. P.-L. Ooi, Y. C. Kuang, and S. Demidenko, ''Multivariate alternating decision trees,'' *Pattern Recognit.*, vol. 50, pp. 195–209, Feb. 2016, doi: 10.1016/j.patcog.2015.08.014.

[64] K. Mansouri, T. Ringsted, D. Ballabio, R. Todeschini, and V. Consonni, ''Quantitative structure–activity relationship models for ready biodegradability of chemicals,'' *J. Chem. Inf. Model.*, vol. 53, no. 4, pp. 867–878, Apr. 2013, doi: 10.1021/ci4000213.

[65] D. D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, and Y. Zhang, ''Failure analysis of parameter-induced simulation crashes in climate models,'' *Geosci. Model Develop.*, vol. 6, no. 4, pp. 1157–1171, Aug. 2013. [Online]. Available: https://gmd.copernicus.org/articles/6/1157/2013/

[66] H. T. Kahraman, S. Sagiroglu, and I. Colak, ''The development of intuitive knowledge classifier and the modeling of domain dependent data,'' *Knowl.-Based Syst.*, vol. 37, pp. 283–295, Jan. 2013, doi: 10.1016/j.knosys.2012.08.009.

[67] A. Tsanas, M. A. Little, C. Fox, and L. O. Ramig, ''Objective automatic assessment of rehabilitative speech treatment in Parkinson's disease,'' *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 1, pp. 181–190, Jan. 2014.

[68] I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror, ''Result analysis of the NIPS 2003 feature selection challenge,'' in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2004, 2004, pp. 545–552. [Online]. Available: https://proceedings.neurips.cc/paper/2004/hash/5e751896e527c862bf67251a%474b3819-Abstract.html

[69] B. E. Sakar, M. E. Isenkul, C. O. Sakar, A. Sertbas, F. Gurgen, S. Delil, H. Apaydin, and O. Kursun, ''Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings,'' *IEEE J. Biomed. Health Inform.*, vol. 17, no. 4, pp. 828–834, Jul. 2013, doi: 10.1109/JBHI.2013.2245674.

[70] M. A. Little, P. E. McSharry, S. J. Roberts, D. A. Costello, and I. M. Moroz, ''Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection,'' *Biomed. Eng. OnLine*, vol. 6, no. 1, p. 23, 2007.

[71] *Statlog (Vehicle Silhouettes) Data Set*, Turing Inst., Glasgow, Scotland, 1987.

[72] N. G. C. F. M. de Abreu, ''Analise do perfil do cliente recheio e desenvolvimento de um sistema promocional,'' Ph.D. dissertation, ISCTE-Univ. Inst. Lisbon, Lisbon, Portugal, 2011.

[73] B. A. Johnson, R. Tateishi, and N. T. Hoan, ''A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees,'' *Int. J. Remote Sens.*, vol. 34, no. 20, pp. 6969–6982, Jun. 2013, doi: 10.1080/01431161.2013.810825.

[74] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, ''Modeling wine preferences by data mining from physicochemical properties,'' *Decision Support Syst.*, vol. 47, no. 4, pp. 547–553, 2009, doi: 10.1016/j.dss.2009.05.016.

[75] K. Fernandes, J. S. Cardoso, and J. Fernandes, ''Transfer learning with partial observability applied to cervical cancer screening,'' in *Pattern Recognition and Image Analysis* (Lecture Notes in Computer Science), vol. 10255, L. A. Alexandre, J. S. Sánchez, and J. M. F. Rodrigues, Eds. Faro, Portugal: Springer, Jun. 2017, pp. 243–250, doi: 10.1007/978-3-319-58838-4_27.

[76] F. Zhou, Q. Claire, and R. D. King, ''Predicting the geographical origin of music,'' in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, R. Kumar, H. Toivonen, J. Pei, J. Z. Huang, and X. Wu, Eds. Shenzhen, China, Dec. 2014, pp. 1115–1120, doi: 10.1109/ICDM.2014.73.

[77] (2010). Semeion, Research Center of Sciences of Communication. *Steel Plates Faults Data Set*. Dataset Provided by Semeion, Research Center of Sciences of Communication, Via Sersale 117, 00128. Rome, Italy. [Online]. Available: https://www.semeion.it

[78] B. Johnson, R. Tateishi, and Z. Xie, ''Using geographically weighted variables for image classification,'' *Remote Sens. Lett.*, vol. 3, no. 6, pp. 491–499, Mar. 2012, doi: 10.1080/01431161.2011.629637.

[79] P. K. Wagner, S. M. Peres, R. C. B. Madeo, C. A. de Moraes Lima, and F. de Almeida Freitas, ''Gesture unit segmentation using spatial-temporal information and machine learning,'' in *Proc. 27th Int. Florida Artif. Intell. Res. Soc. Conf., (FLAIRS)*, May 2014, W. Eberle and C. Boonthum-Denecke, Eds. Pensacola Beach, FL, USA, : AAAI Press, 2014. [Online]. Available: https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS14/paper/view/7787

[80] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles, ''Fifty Years of pulsar candidate selection: From simple filters to a new principled real-time classification approach,'' *Monthly Notices Roy. Astronomical Soc.*, vol. 459, no. 1, pp. 1104–1123, 2016, doi: 10.1093/mnras/stw656.

[81] B. Johnson and Z. Xie, "Classifying a high resolution image of an urban area using super-object information," *ISPRS J. Photogramm. Remote Sens.*, vol. 83, pp. 40–49, Sep. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271613001366

[82] R. Bhatt, "Fuzzy-rough approaches for pattern classification: Hybrid measures, mathematical analysis, feature selection algorithms, decision tree algorithms, neural learning, and applications," in *Decision Tree Algorithms, Neural Learning, and Applications*. Amazon Books, 2017.

LEONARDO CAÑETE-SIFUENTES graduated in computer systems engineering from the Tecnologico de Monterrey, Campus Estado de Mexico, in 2015. He received the M.Sc. degree in computer science from the Tecnologico de Monterrey, in 2018, where he is currently pursuing the Ph.D. degree in computer science. His research interests include supervised classification, interpretable classifiers, and clustering.

RAÚL MONROY received the Ph.D. degree in artificial intelligence from Edinburgh University, in 1998, under the supervision of Prof. Alan Bundy. Since 1985, he has been focusing on computing at the Tecnologico de Monterrey. Since 1998, he has been a member of the CONACYT-SNI National Research System, and received the 3rd Rank. In 2010, he was promoted to a Full Professor in computer science. He is currently the Leader of the Grupo de Investigación con Enfoque Estratégico-Machine Learning (GIEE-ML) Research Group, Tecnologico de Monterrey. His research interests include discovery of novel machine learning models, which he often applies to problems in cybersecurity, automated use of theorem proving to formal methods of system development, discovery an application of general search control strategies for uncovering and correcting errors in either a system or its specification, and discovery of novel motion planning techniques.

MIGUEL ANGEL MEDINA-PÉREZ received the Ph.D. degree in computer science from the National Institute of Astrophysics, Optics and Electronics, Mexico, in 2014. He is currently a Research Professor with the Tecnologico de Monterrey, Campus Estado de Mexico, where he is a member of the Grupo de Investigación con Enfoque Estratégico-Machine Learning (GIEE-ML) Research Group. He received the 1st Rank in the Mexican Research System. He has published ten articles in referenced journals, such as *Information Fusion*, IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, *Pattern Recognition*, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *Knowledge-Based Systems*, *Information Sciences*, and *Expert Systems with Applications*. He has extensive experience in developing software to solve pattern recognition problems. A successful example is a fingerprint and palmprint recognition framework which has more than 1.3 million visits and 135 thousand downloads. His research interests include pattern recognition, data visualization, explainable artificial intelligence, fingerprint recognition, and palmprint recognition.