Information Sciences 583 (2022) 219–238

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

PCTBagging: From inner ensembles to ensembles. A trade-off between discriminating capacity and interpretability

Igor Ibarguren [a], Jesús M. Pérez [b], Javier Muguerza [b], Olatz Arbelaitz [b,*], Ainhoa Yera [b]

[a] Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA), 20500 Arrasate-Mondragón, Spain
[b] Department of Computer Architecture and Technology, University of the Basque Country (UPV/EHU), Manuel Lardizabal 1, 20018 Donostia, Spain

ARTICLE INFO

Article history:
Received 16 March 2021
Received in revised form 9 October 2021
Accepted 5 November 2021
Available online 17 November 2021

Keywords:
Comprehensible classifiers
Interpretable models
Decision trees
Consolidation
Ensembles
C4.5
CTC
Bagging
Machine learning

ABSTRACT

The use of decision trees considerably improves the discriminating capacity of ensemble classifiers. However, this process results in the classifiers no longer being interpretable, although comprehensibility is a desired trait of decision trees. Consolidation (consolidated tree construction algorithm, CTC) was introduced to improve the discriminating capacity of decision trees, whereby a set of samples is used to build the consolidated tree without sacrificing transparency. In this work, PCTBagging is presented as a hybrid approach between bagging and a consolidated tree such that part of the comprehensibility of the consolidated tree is maintained while also improving the discriminating capacity. The consolidated tree is first developed up to a certain point and then typical bagging is performed for each sample. The part of the consolidated tree to be initially developed is configured by setting a consolidation percentage. In this work, 11 different consolidation percentages are considered for PCTBagging to effectively analyse the trade-off between comprehensibility and discriminating capacity. The results of PCTBagging are compared to those of bagging, CTC and C4.5, which serves as the base for all other algorithms. PCTBagging, with a low consolidation percentage, achieves a discriminating capacity similar to that of bagging while maintaining part of the interpretable structure of the consolidated tree. PCTBagging with a consolidation percentage of 100% offers the same comprehensibility as CTC, but achieves a significantly greater discriminating capacity.

1. Introduction

Classification is a machine learning technique used in many fields to build models or classifiers based on training data. Patterns are sought in a set of independent features that determine the value of a dependent feature to correctly predict the value of previously unobserved dependent features in the data. In some of these fields, understanding how the classification is performed is very important, especially in fields where classification is a decision support system for the human user, such as medical diagnosis, law, or fraud detection. As stated in [1], interpretability can be beneficial as an additional design driver for three reasons: to ensure impartiality in decision making, detect, and consequently, correct for bias in the training dataset. It also facilitates robustness by highlighting potential adversarial perturbations that can impact the

* Corresponding author.
E-mail addresses: igor.ibarguren@ikerlan.es (I. Ibarguren), txus.perez@ehu.eus (J.M. Pérez), j.muguerza@ehu.eus (J. Muguerza), olatz.arbelaitz@ehu.eus (O. Arbelaitz), ainhoa.yera@ehu.eus (A. Yera).

https://doi.org/10.1016/j.ins.2021.11.010
0020-0255/

prediction. Finally, interpretability can act as an insurance such that only meaningful variables are used to infer the output, guaranteeing that an underlying truthful causality exists in the model reasoning.

Not all classification algorithms produce models that are interpretable by humans. A widely known and used classification algorithm with explaining capacity, which is transparent according to [1], is the decision tree algorithm. Decision tree models are easily interpretable because they can be graphically displayed as a hierarchy of nodes, where each node creates a path to its child nodes depending on the value(s) of an attribute of the data.

One of the weaknesses of decision tree algorithms is their structure; consequently, the explanation they provide is highly dependent on the training data. Small changes in the data can result in models that are significantly different while still equivalent in their ability to correctly classify data. This property is called instability and can result in users losing confidence in the algorithm [2].

The typical way to improve the results of decision tree algorithms is to create ensemble algorithms. Ensemble algorithms are multiple classifier systems that build several classifiers from the training sample by combining them in some way, such as voting as in bagging [3], to classify new examples. However, as Domingos pointed out, the ensemble voting process obfuscates the reasoning behind the classification because a single set of conditions is no longer related to the values of the variables; there are usually dozens of separate sets of conditions, and even the most patient user cannot handle that [4]. Thus, as stated in [1], tree ensembles increase the generalisation capacity but cause loss of transparency and, as such, are not considered comprehensible.

The consolidation approach was proposed as an alternative to improve the results of decision tree algorithms by maintaining the interpretability of the models [5]. Consolidation also uses multiple samples but creates a single classifier by applying the voting process during the model building phase instead of applying it during the classification of new examples. 'Inner Ensembles' was coined to represent such approaches in [6]. The first algorithm consolidated was the widely known C4.5 decision tree algorithm [7], resulting in the consolidated tree construction (CTC) algorithm. The results demonstrated that not only did consolidation improve the discriminating capacity of C4.5, but also created classifiers that were much more stable [8,9]. However, compared to bagging [3], CTC cannot achieve a low error rate [10,11].

This study proposes a hybrid approach between CTC and bagging, referred to as partially consolidated tree bagging, or PCTBagging. The CTC classifier is developed up to a certain point, and from then on, the tree is developed with each of the samples as in the case of standard bagging. The part of the CTC initially developed is determined by the consolidation percentage, which is a parameterizable value. This way, the comprehensible structure of the consolidated tree is partly maintained, and the bagging part provides an increase in accuracy. PCTBagging represents a trade-off between model interpretability and accuracy. Experiments for 11 consolidation percentages (100%, 90%, 80%, ...10%, and 0%) were performed, and the results were compared with those of bagging (0% consolidation), CTC (100% consolidation), and C4.5 itself. The algorithms were compared in terms of discriminating capacity, the complexity of the explanation provided by the comprehensible algorithms, and the computational cost of building the classifiers. The results demonstrate that PCTBagging with low consolidation percentage achieves a similar discriminating capacity to bagging, while retaining part of the CTC explanation. PCTBagging with high consolidation percentage can also significantly improve the discriminating capacity of CTC without losing interpretability.

The remainder of this paper is organised as follows. In Section 2, insight into previous work is provided. Section 3 explains the PCTBagging proposed in this paper. In Section 4, the experimental setup is presented. In Section 5, the empirical results of the experiments are outlined. Finally, in Section 6, concluding remarks are presented along with suggestions for future work.

2. Background work

This section describes the classification algorithms that serve as the basis for PCTBagging proposed in this paper.

2.1. C4.5

The C4.5 algorithm [7], which is one of the most commonly used transparent algorithms and which is still considered among the top 10 machine learning algorithms[1], was designed by Ross Quinlan to build decision trees. C4.5 was also identified as one of the top 10 algorithms in data mining at the IEEE International Conference on Data Mining held in 2006 [12] and has been widely used in multiple classifier systems since the very beginning [3].

2.2. Bagging

Bagging or bootstrap aggregating [3] is an ensemble algorithm that aggregates multiple models (decision trees, linear regression models, etc.), and the final outcome is obtained by voting among the built classifiers.

Bagging uses the same algorithm, originally C4.5 in [3], to build classifiers with different training samples. These different samples are bootstrap samples (samples of the same size as the original sample, obtained repeatedly selecting examples with replacement), which is the origin of the name bootstrap aggregating.

[1] https://www.kdnuggets.com/2017/10/top-10-machine-learning-algorithms-beginners.html

Ensemble classifiers usually obtain higher model accuracy than their constituent classifiers [13] as long as the individual classifiers that compose the ensemble disagree with each other [14,15]. Tests run by Breiman show a reduction in misclassification rates in the range of 20% to 47%.

A variant of bagging, called subsample aggregating (subbagging) [16], uses samples generated by random undersampling to create individual classifiers instead of bootstrap samples. This variant provided results that were competitive with bagging with a smaller computation cost.

2.3. Consolidation

The consolidation of decision tree algorithms uses the ensemble voting mechanism during the building process of the classifier [17]. Consequently, a single classifier is built to maintain the transparency and interpretability of the base classifier algorithm. Such approaches were dubbed 'Inner Ensembles' and were also applied to Bayesian networks and the k-means clustering algorithm [6].

Consolidation works by first creating multiple samples (*N_S*), as does bagging. CTC originally created stratified samples: samples that retained the class distribution of the original dataset with subsample sizes determined relative to those of the original dataset.

For clarification, CTC could be regarded as a bagging of C4.5 decision trees that are built concurrently but not independently as in bagging. From each subsample, a C4.5 (sub) tree begins to grow, but instead of splitting the subset on its own, each subtree casts a vote for the best split for the particular sample, and all subtrees comply with the majority vote and split their sample accordingly, even if it is not what they originally voted for. If the most voted variable is a discrete variable, a child is created for each possible value, as in C4.5. If the most voted variable is continuous, the algorithm collects all the proposed cut points and selects the median value among them. The process continues until the majority decides not to split. When a consolidated tree has to classify new examples, the average of the class membership probabilities of that leaf node on each subtree is assigned.

In essence, a CTC classifier can be considered a subbagging of C4.5, where all of the individual decision trees share the same structure and node conditions, with each subtree having different class probabilities on their leaves. From an implementation point of view, however, it is not necessary to build all the subtrees. Because all subtrees have the same structure, the actual output model of the algorithm is a decision tree having a structure identical to that of the subtree and leaf class probabilities computed from the probabilities of the subtrees. Thus, being C4.5 interpretable, CTC is also interpretable, even if it combines the knowledge of multiple samples as black-box ensembles.

CTC was subsequently directly compared to bagging and to a post-hoc explainability technique designed to explain the decisions of bagging, referred to as combined multiple models (CMM) [4]. The results of the comparison [10,11] demonstrated that bagging achieves the best discriminating capacity, followed by CTC, with CMM performing worst. Considering the stability of the classifiers, CTC created decision trees that were significantly more stable than CMMs. CTC benefited from changing the domain class distribution during resampling to create multiple samples. Specifically, samples with the optimal class distribution proposed by Weiss and Provost [18] created the best classifying consolidated trees [5]. Instead of using a fixed number of samples, a new resampling strategy was tailored to account for the different degrees of information loss occurring when balancing samples from domains with imbalanced class distributions, such as, coverage-based resampling [19].

Coverage is defined as the percentage of examples of the training sample present in (or covered by) the set of generated (sub) samples. The number of samples required to achieve a specific value of coverage depends on the type of samples used, such as balanced, bootstrap, and stratified (the specific formulations and examples can be found in [20]). As stated in [19], an empirical study determined that 99% is the best coverage value for CTC.

3. Partially consolidated tree bagging

This paper proposes partially consolidated tree bagging (PCTBagging), which is a hybrid approach between CTC and bagging, with objective to maintain the interpretability of CTC while also harnessing the discriminating capacity increase offered by bagging. PCTBagging first creates multiple samples from the original training sample using a resampling strategy determined by the resampling type and number of samples. From these samples, a partially developed CTC tree is created. Subsequently, as many copies of the CTC tree as samples were created are developed independently, similar to during bagging. PCTBagging has to decide when to stop the consolidation process. To analyse the effect of this decision on the results, the consolidation process is stopped at different percentage points in the process of building the complete consolidated tree. Thus, the complete CTC tree is initially built in this work and, once the number of internal nodes is determined, a percentage of them (consolidation percentage) can be selected to stop consolidation, removing the rest, and start with bagging. A percentage of 100% indicates that no internal nodes will be removed (close to CTC), and a percentage of 0% will result in no internal nodes being kept (same as in bagging). This same pattern of experiments can be repeated for every database.

To implement PCTBagging for different consolidation percentages, once the number of nodes to keep is determined, the internal nodes are ordered according to their size, that is, the number of instances covered by each node. Nodes are deleted

according to their size, starting from the smallest ones. Internal nodes are 'removed' by turning them into leaf nodes, thus collapsing the subtree hanging from that node. An algorithmic description of the procedure can be found in Algorithm 1.

Algorithm 1: PCTBagging Algorithm.

Inputs:
S: training set
Criteria: split criteria for building decision trees (C4.5, CHAID...)
N_S (Number_Samples): number of samples to generate
R_M (Resampling_Mode): method used to generate samples
Perc: consolidation percentage
procedure PCTBagging (*S*, *Criteria*, *N_S*, *R_M*, *Perc*)
  $ct :=$ **CTC** $(S, Criteria, N\_S, R\_M)$ // Build consolidated tree
  //$ct \leftarrow$ consolidated tree + *N_S* subtrees
  //Build a list with the internal nodes (not leaf) of *ct* sorted by size
  $LiNodes \leftarrow \{\}$
  $CurrentNode := ct.GetRootNode()$
  while *CurrentNode*!= null **do**
    if *CurrentNode* is not leaf **then**
      Obtain (*id_node*, *size_node*) of *CurrentNode*
      $LiNodes := LiNodes \cup (id\_node, size\_node)$
    end if
    $CurrentNode := ct.NextNodeInPreOrder(CurrentNode)$
  end while
  Sort *LiNodes* by size of node in descending order
  // Remove from *LiNodes* the nodes to maintain as consolidated
  $ConsoPerc := 0$
  while $ConsoPerc < Perc$ **do**
    $LiNodes.DeleteHead()$
    Update *ConsoPerc*
  end while
  for each *CurrentNode* in *LiNodes* **do**
    $ct.Collapse(CurrentNode)$
    // Develop all subtrees independently as Bagging does
    for $i := 1$ to *N_S* **do**
      Let S_i^c be the *i*-th sample according $CurrentNode(c)$ in *ct*
      // Build decision tree according to *Criteria* (C4.5, CHAID...)
      $dt :=$**DecisionTree(**$Criteria, S_i^c$**)**
      Let *subt* be the subtree associated with the *i*-th sample in *ct*
      $subt := subt(CurrentNode) \cup dt$
    end for
  end for
end procedure

Fig. 1 graphically illustrates an example of the process. In the first frame, a fully built consolidated tree is shown, with internal nodes shaded and marked as I_i where $size(I_i) >= size(I_j), \forall i < j$. The figure simulates PCTBagging with a consolidation percentage of 60%. This means that because the consolidated tree shown in the first frame has 10 internal nodes, 6 will be kept. The four nodes to be removed —I_7, I_8, I_9, and I_{10}— are marked with a thick border.

From a structural point of view, a consolidated tree looks like a simple decision tree—C4.5, in this case. However, internally, each consolidated node has *N_S* number of instance subsets, that is, one subset for each of the multiple samples created at the beginning. The last stage of building a PCTBagging classifier consists of growing normally C4.5 trees for each instance subset, as would have been done in bagging. Therefore, *N_S* trees which have identical structures in the consolidated part, differ from this point onward. The classification of new examples is performed in the same way as in bagging. The resulting PCTBagging is illustrated in the second frame of Fig. 1, where there are *N_S* trees that maintain 60% of the structure of the original CTC (shaded internal nodes), and the rest are developed independently from the corresponding sample, analogous to C4.5 (white internal nodes and leaves).

PCTBagging represents a trade-off between the inner ensembles, full-fledged ensembles, CTC, and bagging in this case. The interpretability of CTC (or part of it) is maintained, and the discriminating capacity increase of bagging is also partially harnessed. The consolidation percentage determines how much of the comprehensible model is maintained. A consolidation

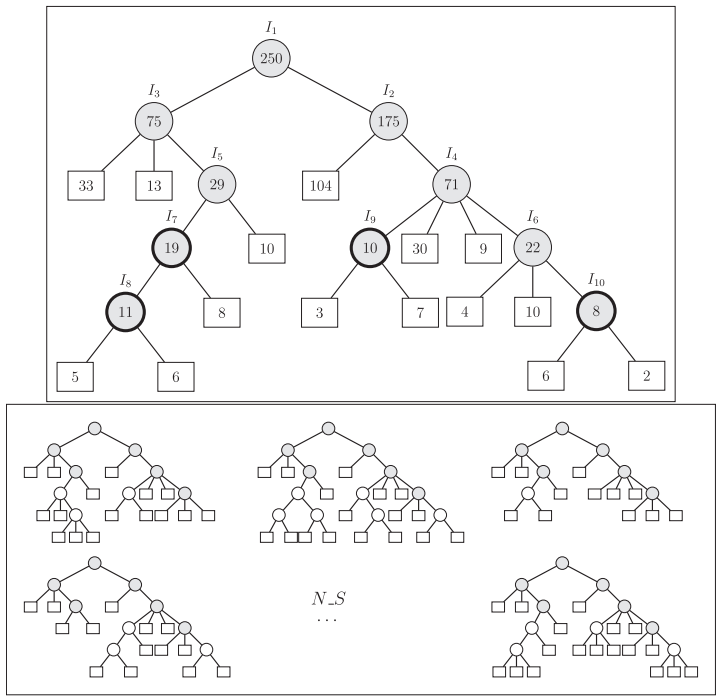


Fig. 1. Partially consolidated tree bagging creation.

percentage of 100% keeps the original consolidated tree intact and grows a bagging on each leaf node. A consolidation percentage of 0%, on the other hand, results in the tree being fully collapsed onto the root, from which a bagging is grown. Using 0% is impractical because it is equivalent to creating a bagging model afresh, which would require about twice as much time. However, low consolidation percentage values can maintain part of the interpretability of CTC while achieving a model accuracy similar to that of bagging.

For documentation and diffusion purposes, PCTBagging was implemented for the popular WEKA framework[2].

4. Experimental setup

The experiments were conducted on 96 datasets from the KEEL repository [21] using a 5-fold 5-run cross-validation. These datasets were originally used in [22] to compare the performance of 16 genetic machine learning (GBML) algorithms for rule induction with six classical nonevolutionary algorithms, such as decision trees, all having explaining capacity. The datasets were divided into three categories. One of the contexts represented classification problems as a whole and included datasets with a wide set of characteristics: two-class and multi-class, small and big datasets, and nominal and continuous variables, as well as a wide range of class distributions. The next context used two-class datasets representing a specific problem within the classification, namely the problem of class imbalance [23]. Finally, the third group used the same context on training sets balanced using the synthetic minority oversampling technique (SMOTE) [24]. In our work, the 96 datasets are grouped as a whole to obtain a global point of view of the performance of the classifiers. A summary of the characteristics of the datasets is provided in Table 1, which includes number of examples (#Examples), number of attributes (#Attributes), number of classes (#Classes), proportion of minority class examples (% Minority Class), and number of examples of the minority class (#Examples Min. Class). Statistics for these characteristics and each of the subgroups of datasets are reported: the minimum, maximum, mean, standard deviation, and median. The full tables detailing each dataset in this article have been moved to additional material[3].

The metric used to assess the performance of the classifiers is the area under the ROC curve (AUC) [25]. AUC evaluates classifiers in multiple contexts of the classification space without assuming any misclassification costs or prior probabilities [26]. AUC is considered a better metric than the overall accuracy (correct classification rate) [27], especially in the presence of class imbalance [28], which is the case for most of the datasets used in this study. Accordingly, in the final analysis, the results include balanced accuracy and sensitivity, two metrics widely used in the context of imbalanced datasets because they focus on the accuracy obtained for single classes, expressed as the average of all classes and the minority class, respectively.

[2] http://www.aldapa.eus/res/weka-pctbagging/
[3] http://www.aldapa.eus/res/pctbagging/

Table 1
Summary description of the 96 datasets.

	#Examples	#Attributes	#Classes	% Minority Class	#Examples Min. Class
Standard 30 datasets					
Min.	80	3	2	0.08	1
Max.	1902	33	22	45.59	668
Mean	638.93	11.77	4.27	21.10	139
Standard deviation	493.55	6.44	3.9	16.41	158.42
Median	521.5	9.5	3	23.06	73
Imbalanced 33 datasets					
Min.	150	3	2	0.77	9
Max.	5472	19	2	35.51	560
Mean	919.94	9.39	2	17.61	120
Standard deviation	1151.99	4.17	0	11.70	132.19
Median	482	8	2	15.48	52
Imbalanced + SMOTE 33 datasets					
Min.	200	3	2	50	100
Max.	9824	19	2	50	4912
Mean	1599.88	9.39	2	50	799.94
Standard deviation	2160.06	4.17	0	0	1080.02
Median	888	8	2	50	444

For more details see: http://www.aldapa.eus/res/pctbagging/

The complexity of the explanation provided by the models is calculated as the number of internal nodes within the decision trees [29]. Bagging is not comprehensible; therefore, it is not calculated. In the case of PCTBagging classifiers, only the consolidated part is considered.

Finally, the computational cost is defined as the time taken to build the classifiers (construction time), measured in milliseconds. All experiments in this work were performed using the same hardware and software. The hardware node had an Intel Core i7-4790 processor (3.60 GHz) and 16 GB of RAM. The operating system was Windows 7 Enterprise SP1, and all the algorithms were developed with Visual C++ within a proprietary platform for algorithms with explaining capacities called *Haritza* (oak in Basque).

In the experiments, the best resampling strategy for bagging and CTC was initially determined. The default recommended configuration for each algorithm is not the same. Bagging, as the name suggests, should be used with bootstrap samples. A fixed number of 50 is widely accepted. In contrast, for CTC, the use of balanced subsamples using 99% coverage to determine the number of samples for each dataset is recommended [19]. The number of samples varied according to the class distribution in the database. The greater the imbalance, the greater the number of samples required. Table 2 displays a summary of the number of samples required for all databases (The number of samples required for each dataset can be found in the additional material). The balanced samples are generated by undersampling the majority classes until the size matches that of the smallest class, to obtain the largest possible balanced subsample without oversampling the minority classes. To explore all the possibilities outside the recommended resampling strategy, two other possibilities were considered for each algorithm: using a set of 50 balanced subsamples and using bootstrap samples with the number of samples determined by the 99% coverage. Only five bootstrap samples were required for any dataset to achieve a coverage value of 99% [20]. Therefore, each algorithm was evaluated with four resampling strategies combining two types of samples (bootstrap and balanced) and two methods to determine the number of samples (fixed to 50 samples and based on 99% coverage). Table 3 displays the abbreviations used for the resampling strategies appearing throughout the paper.

Table 2
Summary of number of balanced samples required to achieve a coverage value of 99%.

Average	Median	Minimum	Maximum	Standard Deviation
53	23	6	585	90.25

Table 3
Abbreviations to refer to the resampling strategies used throughout the paper.

Resampling strategies	Bootstrap		Balanced	
	$N\_S = 50$	Coverage = 99%	$N\_S = 50$	Coverage = 99%
Abbreviation	*boot_N_S*	*boot_cov*	*bal_N_S*	*bal_cov*

After determining the best resampling strategies for bagging and CTC, they were compared to PCTBagging and C4.5. For PCTBagging, 11 consolidation percentage values were considered: 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100%.

4.1. Used statistical procedures

The results were tested using state-of-the-art statistical procedures to determine the significance of the differences. Non-parametric tests have been used with great success for more than a decade since being proposed by Demšar in 2006 in the Journal of Machine Learning Research [30]. At the time of writing this paper, there were 6351 citations in the main collection of the Web of Science. Several studies have complemented the procedure, particularly by adding multiple comparison tests (comparisons involving more than two algorithms) and more powerful post-hoc tests (to be applied to the differences detected over the whole multiple comparison) [31–33] (with 899, 1144, and 2368 citations, respectively, to date). More recently, Benavoli and Corani opted to use Bayesian tests instead of Null Hypothesis Statistical Tests (NHST) [34–36] and published a paper in 2017 in conjunction with Demšar also in the Journal of Machine Learning Research with a very suggestive title: 'Time for a change: a Tutorial for ...'[37]. This paper has 83 citations to date, and at the time it was published the Demšar paper [30] had around 3000 citations (it has had more than 3300 additional citations since then). The evolution of the number of citations of the most relevant papers published during the last years in the context of the analysis of statistically significant differences in classification problems can be found in Fig. 2. The figure includes the reference work [30] preceding that of Demšar, the parametric test proposed by Dietterich in 1998 [38], which is still cited. A critical review of these tests, along with a very interesting practical illustration of their use based on the developed R package *rNPBST*[4] [39], was recently published [40] in the context of [33] by the same research group.

The authors claim that Bayesian tests avoid some of the main pitfalls of NHST. A common mistake in the interpretation of the results of the NHST appears when the null hypothesis is not rejected. The affirmation 'there is no statistical difference between the compared algorithms' is not correct, the true statement should be: 'based on the data, there is not sufficient evidence to reject the null hypothesis'. Bayesian tests compute the distribution of the analysed parameters. The distribution function provides the probability of the null hypothesis being true which is what we usually want to know. Meanwhile, NHST tests actually compute the probability of obtaining a new sample as far from the null hypothesis as the data, assuming that the null hypothesis is true. On the downside, Bayesian tests require a deeper understanding of the underlying statistics and the way to obtain conclusions is not as direct as in NHST, which results in this type of test to being used less often in experimentation.

In [40], the authors state that there is no single way of obtaining conclusions and encourage the joint use of nonparametric and Bayesian tests in experimental studies to obtain a complete perspective on the results.

Following their advice, the graphs in this study were created with the *scmamp* R package[5] [41]. The graphs reflect on the one hand, the ranking obtained by comparing the algorithms based on the Friedman aligned ranks test [32], and on the other hand, whether there are statistical differences between each pair of algorithms based on the powerful post-hoc Bergman–Hommel procedure [33], or, when the number of compared classifiers > 8, based on the Shaffer test [31]). In each graph, the algorithms are the nodes and two nodes are linked if the null hypothesis of being equivalent cannot be rejected, that is, if the behaviour is statistically similar. Therefore, if two nodes are not linked, there are statistically significant differences between the two algorithms. See for instance left of Fig. 4 or Fig. 8.

In the Bayesian paradigm, as in [40], after having rejected the equivalence of all the mean ranks of the algorithms with the Friedman test, the Bayesian signed-rank test is repeatedly performed for every pair of algorithms, summarising the results in figures similar to those in the reference. The Bayesian signed-rank test computes the posterior distribution which describes the distribution of the mean accuracy difference between the two algorithms. With this procedure, the probability of *Algorithm 1* was determined to be better than the probability of *Algorithm 2* (θ_l), the probability that the two algorithms are practically equivalent (*rope*) (θ_e), and the probability that *Algorithm 2* is better than *Algorithm 1* (θ_r). According to [37], in classification, it is sensible to define two classifiers with a mean accuracy difference of smaller than 1% as practically equivalent, and therefore, the authors define a region of practical equivalence, called *rope*. The width of the *rope* region depends on the range of the compared values. In this work, magnitudes with very different value ranges were compared, such as AUC, structural complexity, and construction time. In the case of construction times, values ranged from 0 to 150000 ms. Based on the example given in [40], the region of practical equivalence was defined as $[-10, 10]$ instead of $[-0.01, 0.01]$, as used normally in the classification context. Accordingly, in the case of structural complexity, where values range from 0 to 150 for internal nodes, the region of practical equivalence was defined as $[-1, 1]$.

The table in Fig. 3 displays, for example, the results of the Bayesian signed-rank test applied to the comparison of AUC values for bagging using bootstrap samples with the number of samples fixed to 50 (*boot_N_S*) versus 50 balanced samples (*bal_N_S*). Based on these results, the *rope* region is the hypothesis with the greatest probability. Although not significant, the equivalence of the two algorithms is the most probable situation. The origin of these values can be seen in the *rNPBST* package, which enables plotting the posterior probability density of the parameter, as shown in the chart in Fig. 3. The vertices of the triangle represent the points where the probability of the true location being in this region is 1. The proportion of the

[4] rNPBST, An R package covering non-parametric and Bayesian statistical tests
[5] *scmamp*: Statistical Comparison of Multiple Algorithms in Multiple Problems

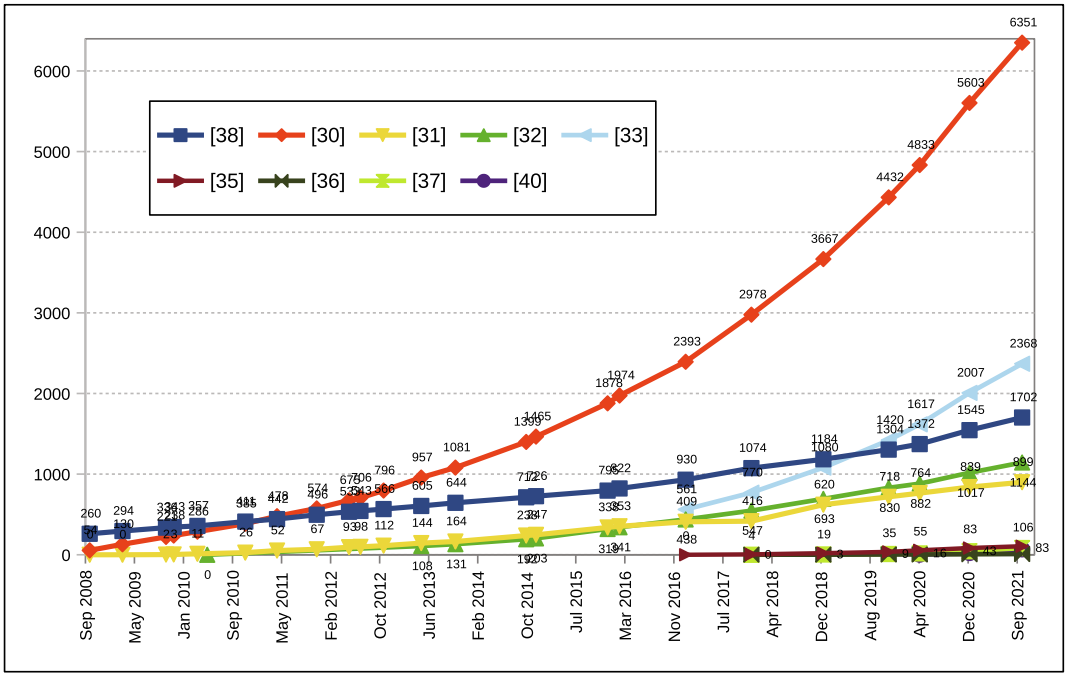
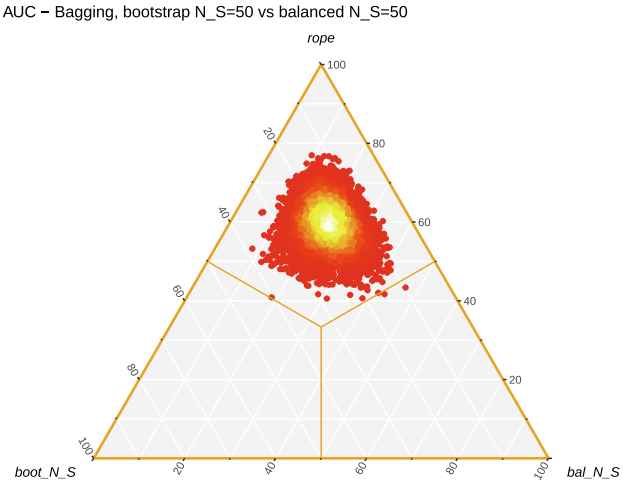


Fig. 2. Evolution of the number of citations of the most relevant papers in the context of the analysis of statistically significant differences.

Bagging	Posterior probability
boot_N_S	0.1869079
rope	0.5944750
bal_N_S	0.2186171

Posterior probabilities



Posterior probability density

Fig. 3. Example of use of the Bayesian signed-rank test on the AUC values for bagging using two different resampling strategies.

locations of the points used to obtain the posterior probabilities is shown in the table in Fig. 3. These tables and figures can be used to counteract the statistical differences between the two algorithms with respect to those obtained on the basis of non-parametric tests. However, in this work, when an $n \times n$ comparison of multiple algorithms is conducted, the results of the tests for each pair of compared algorithms are displayed in figures similar to those presented in Section 9 in [40]. In these figures, only the posterior probability of the most probable option is shown for each pair of algorithms. Each coloured cell represents the comparison of two algorithms, *Algorithm 1* in rows versus *Algorithm 2* in columns. The colour depends on the result of the comparison, indicating that the greater probability is associated with the region of one of the algorithms or with *rope*. The probability of the hypothesis is written in the tile as well as represented by the opacity of the colour, to highlight greater probabilities. In Fig. 4, on the right, is the first example of this type of figure in this paper.

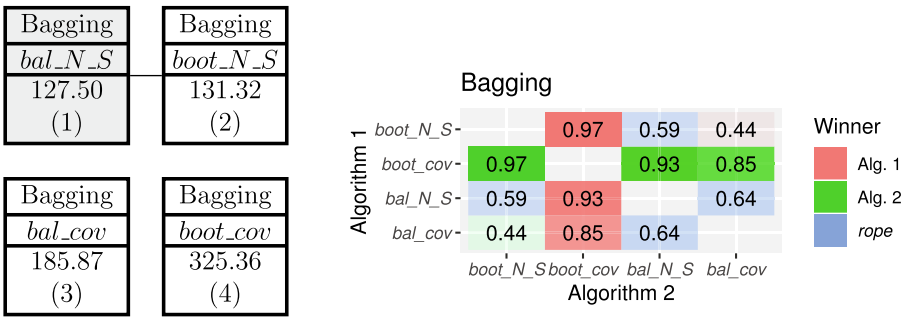


Fig. 4. Statistical tests on the AUC values for bagging with different resampling strategies: Friedman aligned ranks test and the Bergman–Hommel post-hoc procedure (left) and Bayesian signed-rank tests (right).

5. Results

Our goal is to find the best PCTBagging approach and compare it to bagging and CTC. Therefore, the first step was to find the best resampling strategy for these algorithms. Although ideally there would be a single resampling strategy for all, previous results suggest that this is not the case. First, the results for bagging were analysed in terms of discriminating capacity using the four resampling strategies: bootstrap or balanced samples and number of samples set to 50 or based on a 99% coverage. Next, we analyse the results for CTC, and finally, the results for PCTBagging in terms of discriminating capacity, structural complexity, and computational cost. All the previous analyses provided us with information to finally select the most convenient configuration for PCTBagging in each case.

5.1. Results for bagging

Table 4 shows the average values of the area under the roc curve (AUC) obtained for bagging in the 96 datasets. The table also shows the average Friedman aligned rank for each resampling strategy. In this case the average value and the aligned ranks fully agree, indicating that the best strategy is to use a fixed number of 50 balanced samples, closely followed by a fixed number of 50 bootstrap samples and, then, a variable set of balanced samples, different for each dataset, to achieve a coverage of 99%. The results of statistical tests to determine the significance of the differences based on the Bergman–Hommel post-hoc procedure (Fig. 4 left) show that there are no significant differences between PCTBagging and regular bagging using 50 samples. These two perform significantly better than any bagging for the number of samples determined by 99% coverage. In addition, it should be noted that using the coverage combined with balanced subsamples still yields significantly better performance than combining coverage with bootstrap samples.

The results of the pairwise Bayesian signed-rank tests comparing all possibilities are displayed in Fig. 4. Evidently, the results are against the use of bootstrap samples; based on 99% coverage (*boot_cov*) in comparison to the other three options, the probability is worse by 85%. On the other hand, using 50 balanced samples (*bal_N_S*), the first option based on the Friedman aligned rank test tends to be practically equivalent (*rope*) to the other two, (*boot_N_S* and *bal_cov*), although not with high significance. Based on the Bayesian tests, *boot_N_S* seems to be somewhat better than *bal_N_S* when compared to the other two options. However, in terms of computational cost, *bal_N_S* is much less expensive than *boot_N_S*, with average execution time of 1567.35 ms and 7381.91 ms, respectively, and with similar performance in AUC, considering all 96 datasets. Table 5 displays the results of the Bayesian signed-rank test applied to the time values used to build the bagging classifiers for these two types of resampling. The values indicate that using balanced samples is faster than using bootstrap samples with a probability of 93%.

In summary, based on AUC, the best choice for bagging is to use a set of 50 samples instead of the number of samples determined by the coverage; however, the differences with respect to the different sample types were not found to be significant, although using balanced subsamples achieved better average AUC and rank. In addition, with respect to the time it took to build the model, using balanced samples was significantly faster than using bootstrap samples. When both baggings

Table 4
AUC values for bagging using different resampling strategies.

Bagging	Bootstrap		Balanced	
	N_S = 50	Coverage = 99%	*N_S* = 50	Coverage = 99%
Average	.9064	.8576	.9101	.9019
	(2)	(4)	(1)	(3)
FriedmanAligned Rank	131.32	325.36	127.45	185.87
	(2)	(4)	(1)	(3)

Table 5
Posterior probabilities of Bayesian signed-rank test on the time values for bagging using two different resampling strategies.

Bagging	Posterior probability
boot_N_S	0.05090338
rope	0.01622566
bal_N_S	0.93287096

used the same number of samples, the construction time was mostly determined by the size of the samples. Bootstrap samples are always the same size as the original sample, and balanced samples are always smaller than the original sample (on average, 43% of the original sample in the datasets considered). In fact, the more imbalanced the dataset, the smaller the size of the balanced samples. Consequently, it is expected that bagging would work faster using balanced subsamples instead of bootstrap samples. Therefore, because both options are practically equivalent in terms of discriminating capacity, we selected the bagging built with 50 balanced samples. Previous studies have already demonstrated the potential of these ensembles, called subbagging [16,42,43], compared to bagging.

5.2. Results for CTC

The results in Table 6 demonstrate that the best resampling strategy for CTC is to use balanced subsamples and to determine the number of samples according to coverage, followed by a fixed set of 50 for balanced subsamples; the same order is repeated for bootstrap samples. Both the average AUC value and the Friedman aligned ranks concur in this order. As observed in Fig. 5, according to the Bergman–Hommel procedure, all resampling strategies are found to be significantly different from other strategies. Therefore, based on nonparametric tests, it can be stated that using balanced subsamples with the number of samples determined by 99% coverage achieves a significantly better AUC value than any of the other three strategies considered.

In Fig, 5, on the right, the results obtained with the pairwise Bayesian signed-rank tests are displayed. The conservative nature of these tests stands out. There is one strategy that is clearly worse than the other three, using bootstrap with a fixed number of samples, *boot_N_S*, with probabilities ranging from 63% to 79%. Balanced samples based on coverage, *bal_cov*, is the strategy having the greatest advantage over bootstrap and is practically equivalent (*rope*) to using a fixed number of balanced samples.

As was pointed out at the beginning of this section, the ideal situation was expected to be the same resampling strategy for both bagging and CTC. However, a different strategy emerged for each algorithm. For bagging, it is better to use a fixed set of 50 balanced samples, whereas, for CTC, it is better to use a different number of balanced samples for each dataset, as determined by the 99% coverage value.

Table 6
AUC values for CTC using different resampling strategies.

CTC	Bootstrap		Balanced	
	N_S = 50	Coverage = 99%	*N_S* = 50	Coverage = 99%
Average	.8069	.8246	.8316	.8379
	(4)	(3)	(2)	(1)
FriedmanAligned Rank	281.41	197.56	165.04	125.98
	(4)	(3)	(2)	(1)

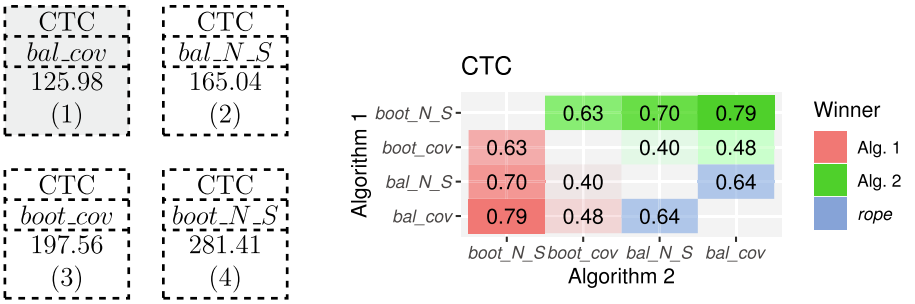


Fig. 5. Statistical tests on the AUC values for CTC with different resampling strategies: Friedman aligned ranks test and the Bergman–Hommel post-hoc procedure (left) and Bayesian signed-rank tests (right).

5.3. Discriminating capacity of PCTBagging

To determine the best resampling strategy for PCTBagging, the results of the best resampling strategies for bagging and CTC were analysed.

Fig. 6 displays the AUC values obtained using the two resampling strategies, for bagging (dotted lines), 11 variations of PCTBagging (continuous lines), CTC (dashed lines), and C4.5 (continuous lines without markers) built using the full training sample. In the figure, lines with triangular markers show the values obtained using a fixed number of 50 samples (best resampling for bagging), and lines with circular markers show the values obtained using the number of samples determined by the coverage (best resampling for CTC).

As expected, in Fig. 6 it is observed that bagging achieves better AUC than CTC, and PCTBagging forms a curve between CTC and Bagging. Ideally, this curve would be vertically mirrored, with the greatest discriminating capacity being gained from the initial percentage values, sacrificing only little of the explaining capacity for a significant increase in AUC. On the other hand, CTC, even with suboptimal resampling, achieves better AUC than C4.5. With regard to PCTBagging, at a consolidation percentage of 100%, the optimal resampling strategy could be expected to be identical to that of CTC because CTC shares the same tree with PCTBagging. Then, at some consolidation percentage, the curves would cross, making the optimal resampling strategy for PCTBagging identical to that of bagging as the percentage gets closer to 0%. The resampling strategy for bagging with 50 balanced samples is preferred to all consolidation percentages for PCTBagging, and therefore, it is selected as representative of this algorithm. The values of the two curves display a strictly monotonical increase as the consolidation percentage decreases. However, even if the gap between the curves increases when the consolidation percentage approaches 0%, the results for the pairwise Bayesian signed-rank tests in Fig. 7 show that based on AUC, for the same consolidation percentage, the behaviour for the two resampling strategies is practically equivalent. The gap between CTC and PCTBagging when the consolidation percentage is 100% and the same samples are used is notable. These values could be expected to be closer; however, it seems that the effect of allowing the trees to develop from the leaf nodes of the consolidated tree, as in bagging, has more weight: that is, it is more relevant than sharing 100% of the structure of the CTC.

AUC	C4.5	CTC	PCTBagging											Bagging
			100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%	
bal_N_S	.8281	.8316	.8711	.8722	.8741	.8766	.8795	.8836	.8868	.8910	.8960	.9025	.9101	.9101
bal_cov		.8379	.8682	.8692	.8707	.8729	.8750	.8788	.8812	.8849	.8891	.8953	.9019	.9019

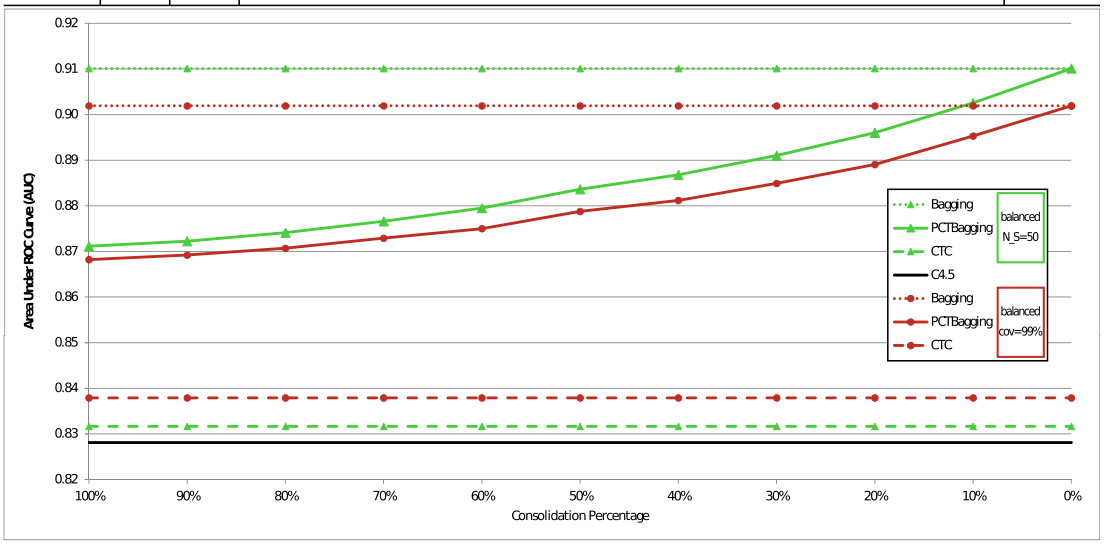


Fig. 6. Average AUC values for all algorithms using two resampling strategies with balanced subsamples.

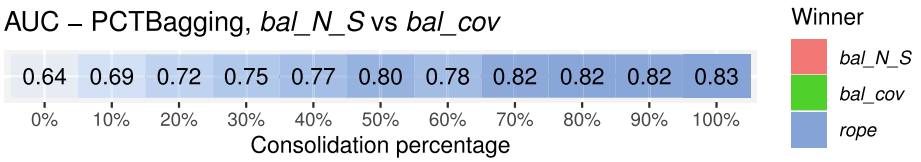


Fig. 7. Bayesian signed-rank tests on the AUC values for PCTBagging using balanced samples to compare the use of a fixed value of 50 samples and a variable number of samples determined by the 99% coverage value.

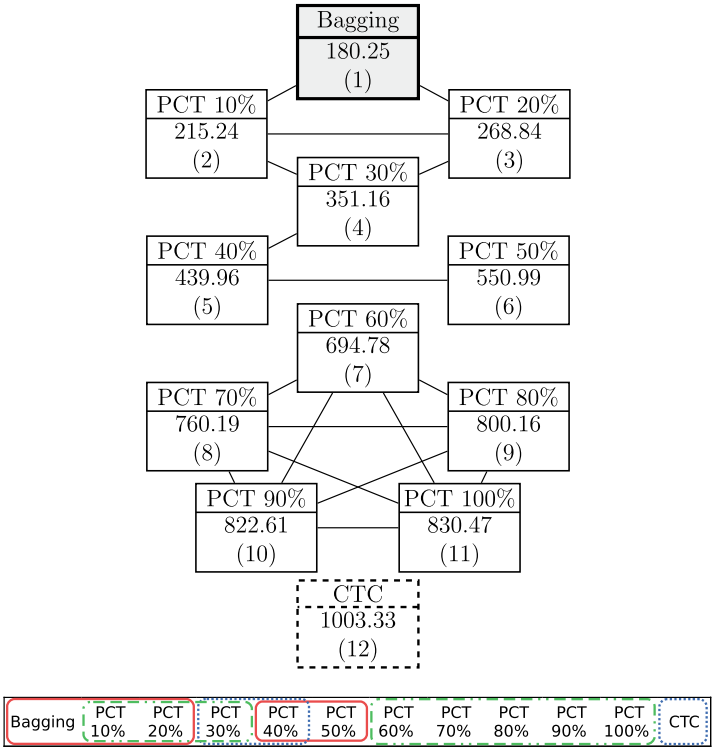


Fig. 8. Friedman aligned ranks and pair-wise p-values on the AUC values for bagging, PCTBaggings and CTC.

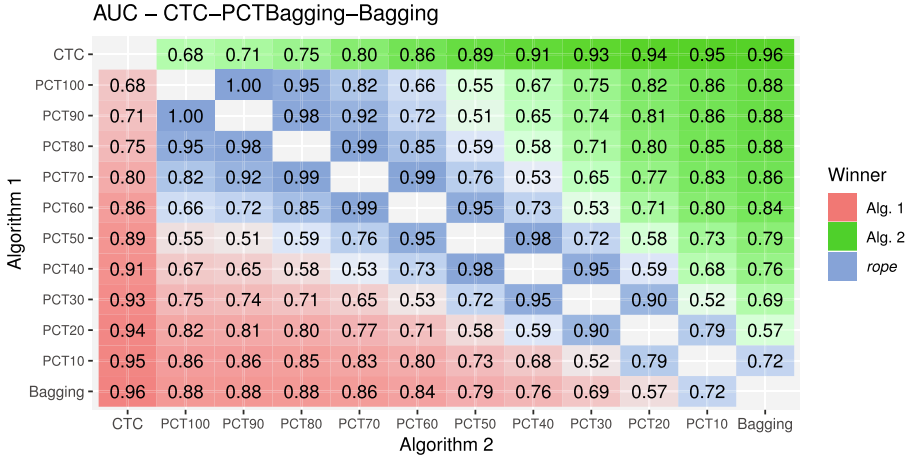


Fig. 9. Bayesian signed-rank tests on the AUC values for bagging, PCTBagging and CTC.

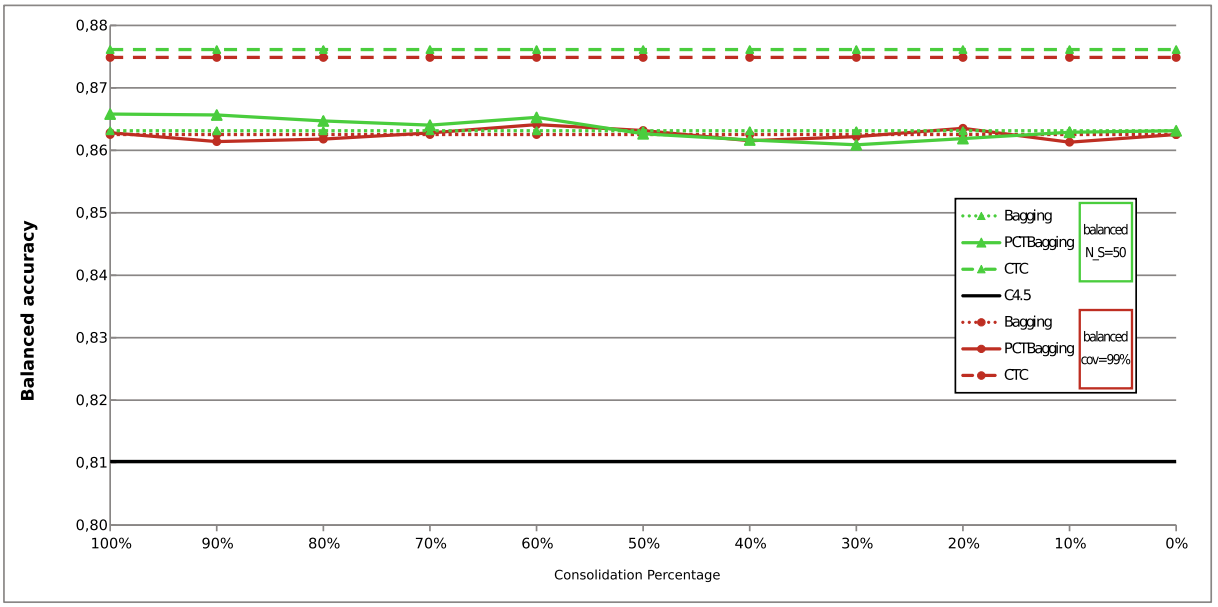
Once the best resampling strategies for bagging, CTC, and PCTBagging have been determined, they are subjected to an $n \times n$ comparison to statistically determine whether the differences between the algorithms is significant. Fig. 8 displays the Friedman aligned ranks and the pairwise differences determined by the Shaffer procedure[6], comparing bagging and PCTBagging with 10 different percentages using 50 balanced samples and CTC using a number of samples based on coverage. In addition, in the table below, the graph is further simplified: algorithms with similar behaviour appear in the same group. This figure shows that the differences between bagging, PCTBagging at 10%, and PCTBagging at 20% are not significant. PCTBagging at 30% already performs significantly worse than bagging. The rest of the PCTBagging options also perform significantly worse than bagging, and no differences are found between PCTBaggings with consolidation percentages between 60% and 100%. Finally, CTC performs worse than all PCTBagging options (even at 100%).

[6] used instead of Bergman–Hommel because the number of classifiers is greater than 8

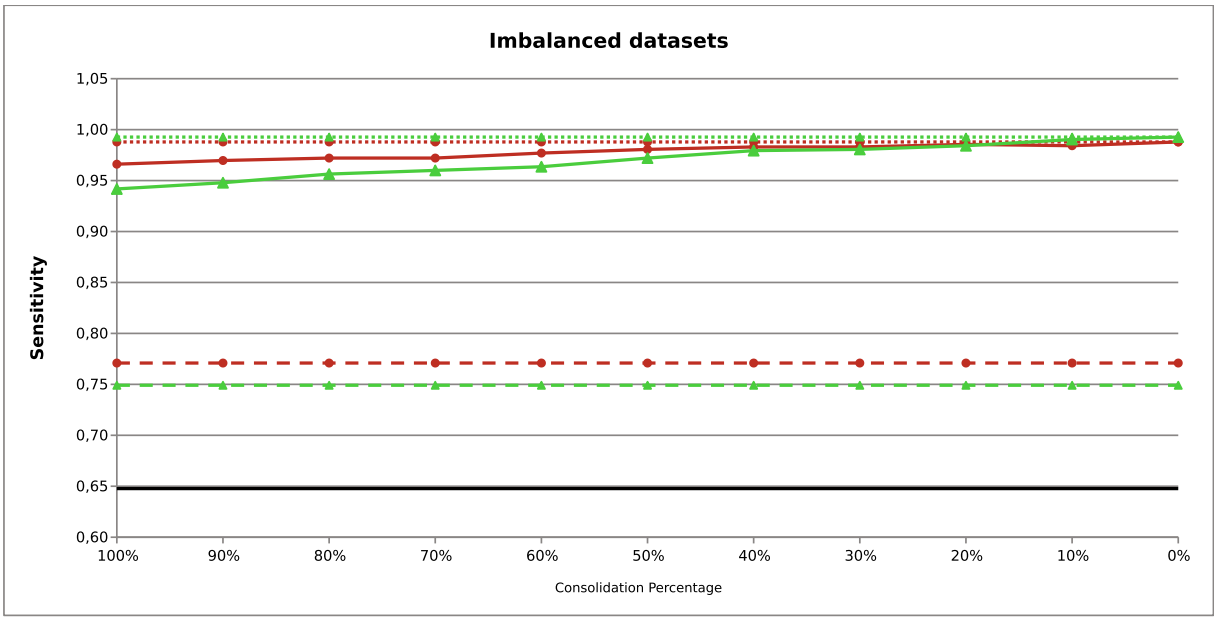
Based on the Bayesian tests results in Fig. 9, the behaviour of bagging is similar only to PCTBagging with a 10% consolidation percentage. From 20% consolidation, bagging is initially better, with low probability (57%), and with values above 80% probability when the 60% consolidation percentage is exceeded. The figure shows that all PCTBaggings with a consolidation percentage between 60% and 100% have practically equivalent behaviour (*rope*) (the curve in Fig. 6 is less steep in this range), similar to the results of the nonparametric tests.

5.3.1. Balanced accuracy and sensitivity

To confirm the conclusions inferred from AUC, a balanced accuracy analysis was also included for the 96 datasets and a sensitivity analysis was conducted for the imbalanced datasets. The results are shown in Figs. 10a and 10b. In the case of balanced accuracy, CTC, classifies all individual classes better, whereas PCTBagging performance is equivalent to bagging



(a) Balanced accuracy values for all datasets.



(b) Sensitivity values for imbalanced datasets.

Fig. 10. Average balanced accuracy and sensitivity values for all algorithms using the two resampling strategies with balanced subsamples.

for every percentage. When assessing the classification of the minority class (sensitivity), the previous trends displayed by AUC are confirmed.

5.4. Structural complexity of PCTBagging

Fig. 11 displays the number of internal nodes for CTC, for the 11 PCTBagging options, and for C4.5. With this metric, the goal is to measure the complexity of the classifiers, not the overall complexity. To explore the trade-off between classification accuracy and interpretability, the focus has been only on the complexity of the transparent part of the classifiers. Bagging is not considered interpretable because all the trees tend to be very structurally different and, therefore, the classifiers lose every transparent property [1]. Consequently, the internal node values for bagging are not provided and for PCTBagging, only the consolidated portion of the models is considered. Results show an almost linear relationship for the consolidated nodes of PCTBagging as a function of consolidation percentage extending from 100% (i.e. the entire CTC tree) down to 0% for zero internal nodes. This could be expected because the number of internal consolidated nodes that are maintained during consolidation correspond to the displayed percentage.

The need to statistically test the differences between different consolidation percentages, using the same resampling strategy, is questionable because the number of internal nodes in PCTBagging is not a 'natural' occurrence. Starting from a baseline (the full CTC tree), the internal nodes are forcefully removed from one consolidation percentage to the next, and a specific consolidation percentage always has the same or a lower number of internal nodes than a greater consolidation percentage. It is of interest, however, to know for which values of consolidation percentage (close to 100%) the PCTBaggings built with 50 balanced samples are similar to the CTC built with balanced samples, with the number of samples being based on coverage. Fig. 12 displays the results of the Friedman aligned ranks and the pairwise differences based on the Shaffer procedure, comparing CTC and PCTBagging with 11 different percentages. The results demonstrate that CTC, the second most complex option, displayed no statistically significant difference for the consolidation percentage values of PCTBagging within the range of 70% to 100%. Symmetrically, the same holds true for the lowest consolidation percentage values of PCTBagging, between 0% and 30%. There were significant differences between two large sets of classifiers: those with a consolidation percentage above 50% (including CTC) versus those with a consolidation percentage below 50%. In this case, the pairwise Bayesian tests (Fig. 13) demonstrate that PCTBaggings with a lower consolidation percentage value are simpler by more than 95% probability in most cases. However, CTC is simpler than PCTBagging 100%, equivalent to PCTBagging 90%, and is more complex than PCTBagging 80% (although with only 52% probability).

When comparing the number of internal nodes for the same consolidation percentage and the two resampling strategies considered, the results in Fig. 11 demonstrate that the use of coverage to set the number of samples results in simpler explanations with fewer internal nodes. The results of the pairwise Bayesian tests in Fig. 14 demonstrate that classifiers built with

Internal Nodes	C4.5	CTC	PCTBagging										
			100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
bal_N_S	19.35	21.81	21.81	19.60	17.45	15.23	13.09	10.65	8.72	6.48	4.36	2.11	0.00
bal_cov		18.97	18.97	17.05	15.17	13.23	11.39	9.24	7.57	5.64	3.79	1.81	0.00

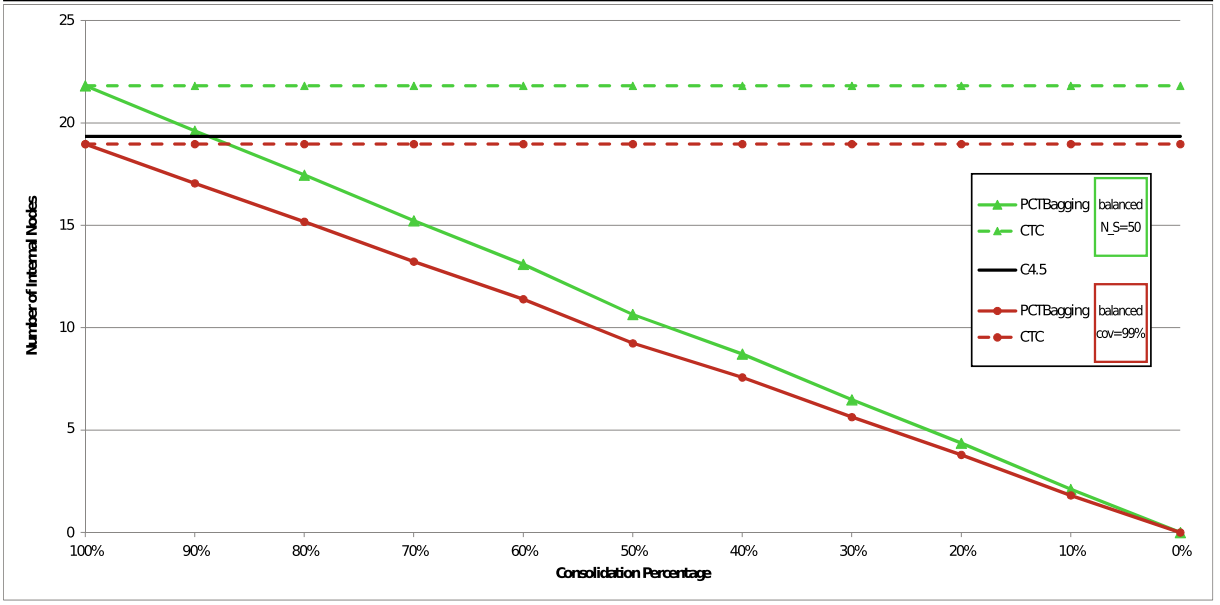


Fig. 11. Average Internal Node values for all algorithms using two resampling strategies with balanced subsamples.

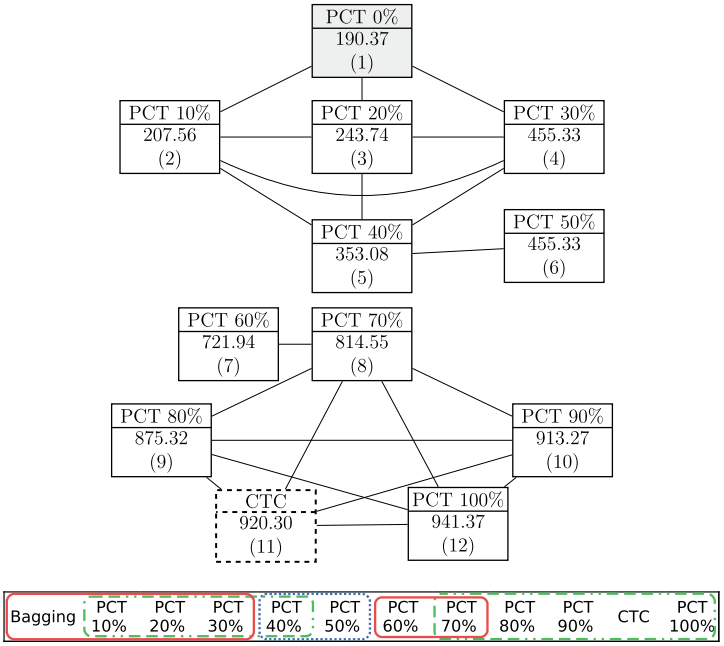


Fig. 12. Friedman aligned ranks and pair-wise p-values for numbers of internal nodes between PCTBaggings and CTC.

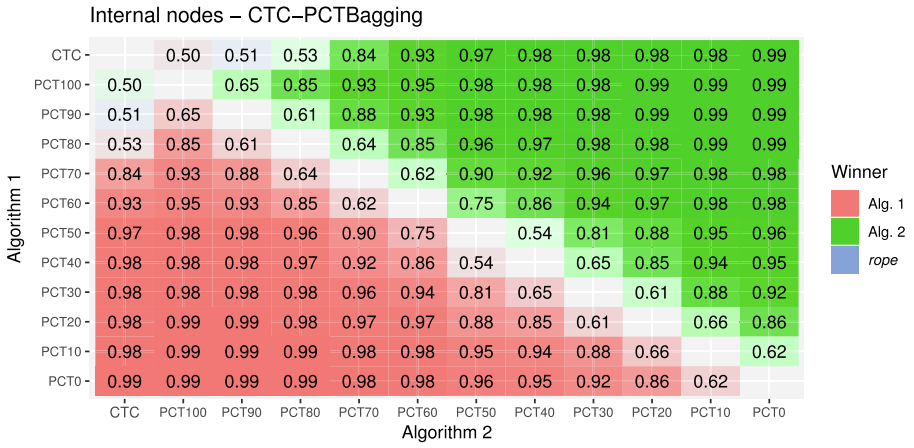


Fig. 13. Bayesian signed-rank tests on the numbers of internal nodes for PCTBagging and CTC.

two types of samples are equivalent (*rope*), with 100% probability for the 0% consolidation percentage (with no internal nodes, regardless of the number of samples). This probability decreases as consolidation percentage increases such that the classifiers built with balanced samples based on 99% coverage, *bal_cov*, are simpler (for consolidation percentages over 90%).

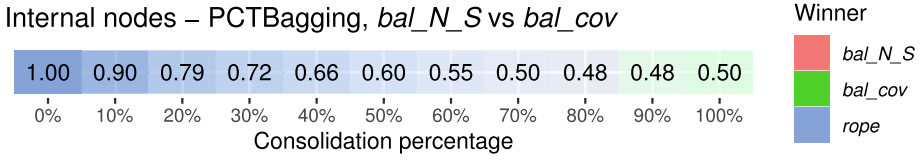


Fig. 14. Bayesian signed-rank tests on internal nodes for PCTBagging using balanced samples; the comparison uses a fixed value of 50 samples or a coverage value of 99%.

5.5. Computational cost of PCTBagging

The results for the average times required to build the classifiers are displayed in Fig. 15. It is observed that using 50 balanced samples takes much more time than determining the number of samples to be used by the coverage. This can be attributed to the same explanation provided for the number of internal nodes: the median *N_S* value determined by the coverage is 23, and the number of samples used (considering all samples are equal in size) is directly related to the time needed to build the consolidated or bagging models. Fig. 16 shows that, based on the pairwise Bayesian signed-rank tests between the two types of resamplings, the construction time of the PCTBaggings based on coverage is lower for all values of consolidation percentage with probabilities above 80%. As expected, the base decision tree algorithm, C4.5, is the fastest. Among the ensemble methods, bagging is the fastest, followed by CTC for every resampling strategy.

The gap between bagging and CTC can mostly be attributed to the construction process. While bagging independently creates each model, stopping each individual classifier as called by the sample, CTC forces subtrees, which would otherwise stop, to keep splitting depending on the majority vote. Another contributing factor, albeit to a lesser degree, is the fact that CTC requires some time to compile each individually proposed split by the subtrees and to compute the consolidated split. A more in-depth analysis of the computational cost of bagging and CTC was conducted in [11].

In the implementation in this work, PCTBagging for any percentage first requires the full CTC tree to be built, and thus, the construction time for PCTBagging is greater than that for CTC.

A pattern is observed in the figure: the time it takes for higher consolidation percentages is more or less the same, with a slight increase when reaching 10% and a greater increase when approaching 0%. Interestingly, based on the figure, it is impossible to determine how the time required for the built varies for higher consolidation percentages. However, the results displayed in the table of Fig. 15, show that for *N_S* = 50, with the exception of 90%, it takes an average of 3 ms more for percentages higher than 80%, whereas the rest of the slope is upward, with increasing differences from 40% onward. In

Time	C4.5	CTC	PCTBagging											Bagging
			100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%	
bal_N_S	224	4366	4407	4559	4556	4556	4558	4575	4609	4668	4751	4895	5416	1567
bal_cov		924	958	1011	1008	1006	1005	1006	1010	1019	1032	1065	1193	627

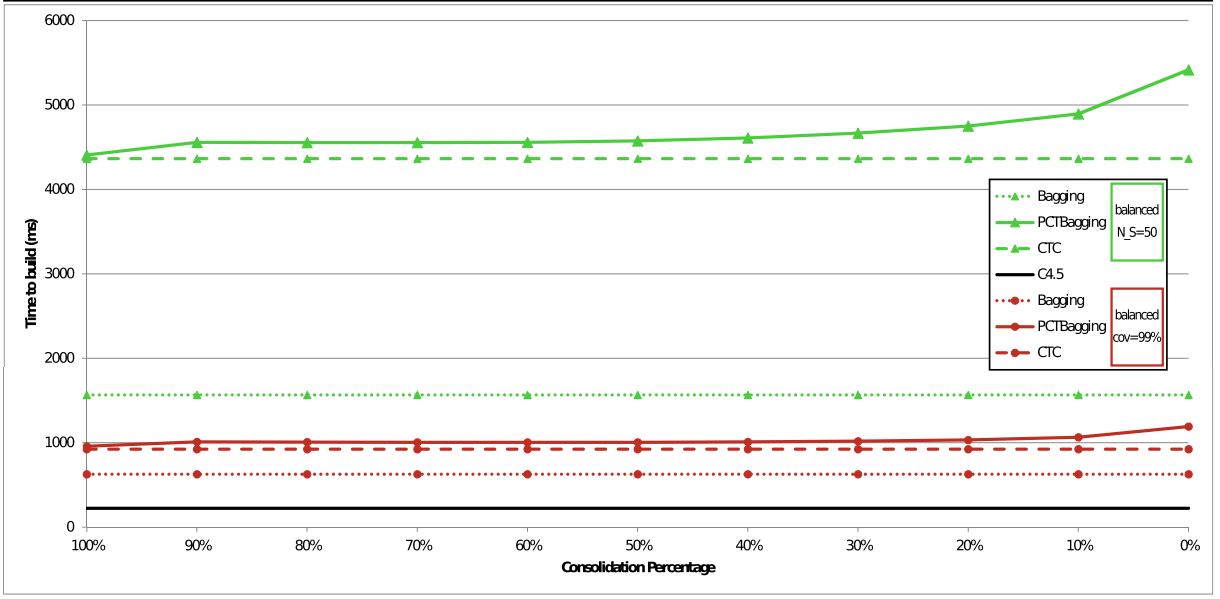


Fig. 15. Average construction time values for all algorithms using two resampling strategies with balanced subsamples.

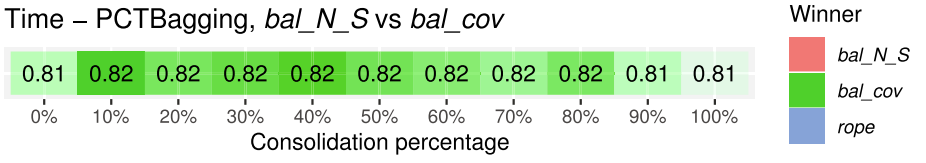


Fig. 16. Bayesian signed-rank tests on construction time for PCTBagging using balanced samples, comparing the use of a fixed value of 50 samples and a coverage value of 99%.

contrast, determining the number of samples by coverage leads to much more stable construction times until a consolidation percentage of 10% is reached, and even then, the differences are relatively small compared to using 50 subsamples. In both cases, a consolidation percentage of 10% requires a relative increase of 11% in construction time compared to a consolidation percentage of 100%.

Fig. 17 shows the results of the Friedman aligned ranks and Shaffer tests for statistical significance using construction time as metric. Bagging and PCTBagging use 50 balanced samples, and CTC uses a variable number of samples based on 99% coverage. In this case, PCTBagging at 0% is also included because even if the final classifiers are the same as in bagging, from a construction point of view, they are not. Although the diagram may look a little intricate, it is easy to interpret. CTC is significantly faster than any other algorithm. There is no difference between PCTBagging at 100%, the next fastest algorithm, and bagging, which, while placing third, is significantly faster than any other PCTBagging. Bagging is significantly faster than any PCTBagging except for 60% consolidation, and the differences among the rest of PCTBaggings are not found to be significant. The ranking position of bagging is surprising based on the average construction times for bagging and PCTBagging 100%, given as 1567 and 4407 ms, respectively in the table of Fig. 15. This can be explained by an analysis of the additional material in the table displaying the times to build the classifiers for the 96 datasets. The values vary greatly from one dataset to another, with the median for bagging being 689 ms and values ranging from 500 to 653 ms for PCTBagging from 100% to 0% consolidation percentage. The median suggests that bagging would be more costly than any of the PCTBaggings (contrary to what the average suggests). With the basic Friedman test proposed by Demšar in 2006 in [30] and even for its Bayesian version proposed in 2017 in [36], the two first positions in the ranking are maintained by CTC and PCTBagging 100%; PCTBagging 60% becomes third (instead of fourth), and bagging falls to ninth position. The Bayesian test results displayed in Fig. 18 confirm the conclusions drawn from the nonparametric tests, where CTC is noticeably less expensive wih respect to time than the rest of the options with probabilities greater than 83%; PCTBagging 100% is also less expensive than the rest, except for CTC and bagging. The same occurs with bagging, although with lower probability values. A change in PCTBaggings between 50% and 90% can also be confirmed where the most probable option becomes the equivalence of classifiers (*rope*).

5.6. Selection of the final PCTBagging based on the consolidation percentage

Taking into account all the results, the choice of the consolidation percentage to be used becomes a matter of preference for the end user. Users weigh the three measures, discriminating capacity, interpretability, and construction time differently based on their interests and needs, but most agree that discriminating capacity is the most important. Considering interpretability, speed, and construction time, it should be noted that building each individual classifier does not take much time—5 s on average using the datasets of this study. Therefore, interpretability is presumably the second most-valued fea-

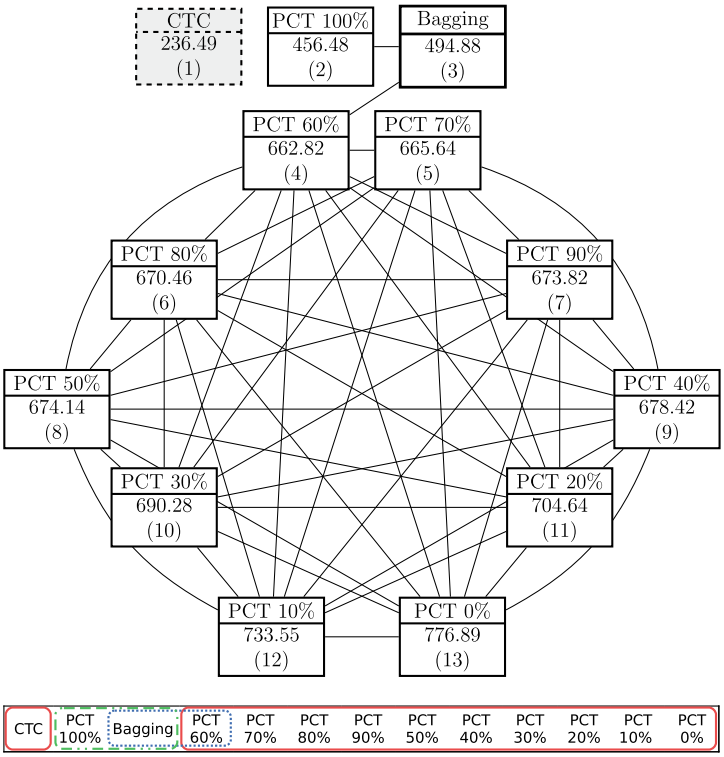


Fig. 17. Friedman aligned ranks and pair-wise p-values for construction time between bagging, PCTBaggings and CTC.

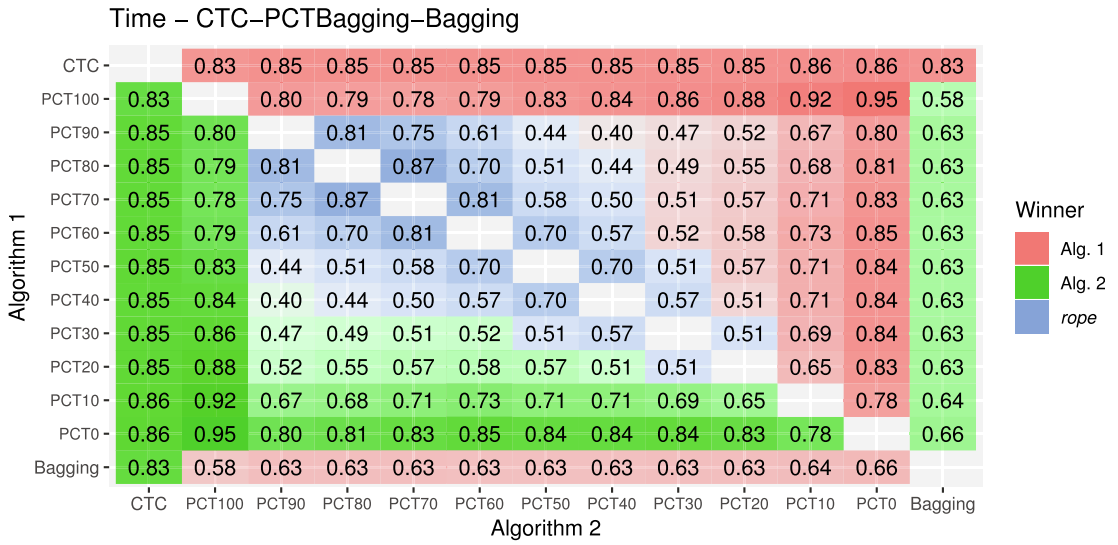


Fig. 18. Bayesian signed-rank tests on construction times for bagging, PCTBagging and CTC.

Table 7
Results for PCTBagging with consolidation percentage of 20 % in all the analysed aspects compared to the rest of the algorithms.

	C4.5	CTC	PCTBagging %20	Bagging
AUC	0.8281	0.8371	0.8960	0.9101
Balanced Acc.	0.8102	0.8749	0.8619	0.8631
Sensitivity	0.6477	0.7709	0.9842	0.9927
Internal nodes	19.35	18.97	4.36	—
Construction time	224	924	4751	1567

ture of the three, with speed being the last. The best option may be a trade-off solution that offers a good level of performance with a certain level of explanation, without neglecting the computational cost.

If performance is the most important consideration, bagging achieves the best average value and rank. However, the two lowest consolidation percentages for PCTBagging achieve a performance that is not significantly worse than that of bagging (based on nonparametric tests). Sacrificing this insignificant amount of AUC would allow the user to keep up to 20% of the consolidated tree comprehensible structure. This means that, on average, 4.36 internal nodes of the consolidated decision tree would be maintained according to the results in the table of Fig. 11. Table 7 illustrates this with a concrete example showing values for all metrics and algorithms. This should be more than sufficient to orient the user to how the classification is done.

Based on the Bayesian tests, considering that the probability of bagging is not any better than 80%, the consolidation percentage can be raised to 40%-50%, which is close to half of the tree resulting in an average of 8.72–10.65 internal nodes. Certainly, some discriminating capacity would be sacrificed in bagging, but considerable interpretability would be gained.

If interpretability is essential, PCTBagging offers an alternative to CTC. At a consolidation percentage of 100%, PCTBagging offers the same comprehensibility but a significantly better accuracy, albeit at a significantly higher computational cost, and it is still computationally less expensive than any other PCTBagging.

6. Conclusions and future work

Consolidation was proposed as an alternative to bagging to improve the performance of decision trees using techniques from multiple classifier systems while maintaining interpretability. Bagging is an ensemble algorithm that creates multiple samples, and a classifier is generated from each sample. The model classifies new examples by putting them through each independent classifier and assigns the class voted by the majority of the classifiers. The multiclassifier nature of bagging leads to a loss of transparency in decision trees. However, interpretability is a desirable trait in many classification domains. When decisions derived from such systems affect human lives, there is an emerging need to understand how such decisions are made. Consolidation follows the 'Inner Ensemble' approach. Multiple samples are created, but the voting process is performed during the classifier building phase, such that the knowledge of multiple samples is used to create a single transparent classifier. CTC, the consolidated version of C4.5, achieved better accuracy and stability while maintaining the interpretability of the decision tree. However, CTC has never matched bagging in terms of discriminating capacity.

In this work, PCTBagging is presented as an approach that sacrifices part of the CTC transparency to obtain a greater classification performance—closer to bagging. This is done based on a set of samples, by building a consolidated tree up to a certain point, that is, partially, and then developing the trees associated with each sample independently, as in bagging. To evaluate the

performance of these classifiers according to the degree of development of the consolidated tree, in this work, a whole CTC tree was initially built. A percentage of internal nodes was then removed, and finally, a bagging model was created by building a C4.5 tree from each subset of instances found on the leaf. The consolidation percentage can be adjusted, and in this study, 11 possible values from 0% (bagging) to 100% (CTC) were used for consolidation. These 11 PCTBaggings were compared to bagging, CTC, and C4.5 (the base classifier) from three points of view: discriminating capacity, structural complexity, and computational cost. In addition, different resampling strategies were analysed for these algorithms to realise the best potential for each option. Balanced samples (instead of bootstrap), with the number of samples fixed to 50, were selected for bagging and PCTBagging. For CTC, balanced samples, with the number of samples determined by 99% coverage, was selected.

The results demonstrated that PCTBaggings with a low consolidation percentage (10% and 20%) achieved a similar AUC compared to bagging while keeping, on average, up to 4.36 internal nodes from the consolidated tree. Increasing the percentage of consolidation gradually led to a decrease in the classification capacity. The highest consolidation percentage (100%) offers the same comprehensible explanation as CTC but achieves a significantly higher AUC than CTC. Accordingly, the decision of determining the consolidation percentage is left to the user to weigh the trade-off between accuracy and interpretability. The lower the consolidation percentage, the greater the discriminating capacity is, but the lower the explaining capacity, bringing it closer to bagging; conversely, the higher the consolidation percentage, the lower the discriminating capacity and the higher the explaining capacity, bringing it closer to a consolidated tree.

In future work, we would first like to develop hybrid versions of PCTBagging. The current implementation creates a C4.5 consolidated decision tree, then removes part of the tree, and creates a bagging of C4.5 decision trees on each node with the subsets (one for each sample) that are assigned to that node (now turned into a leaf). However, there is no need for the bagging to use the same decision tree algorithm, it could be a bagging of any other algorithm. Taking this idea even further, the model grown from the leaves does not have to be an ensemble algorithm: a simple classifier can be built by combining the multiple subsets on the leaves into a single subset. Approaches such as the logistic model tree (LMT) [44] already do this. LMT first builds a simple decision tree and then creates a logistic regression model for each leaf. Another possibility could be to use the naïve Bayes algorithm on the leaves, as already proposed by Kohavi with the NBTree algorithm [45].

If, after partially creating a consolidated tree, we use any other strategy to build ensemble classifiers instead of using bagging with the set of samples in the leaves of the tree, other alternatives can be realised. Algorithms such as boosting [46] and random forests [47] have been shown to be great alternatives to bagging [48,49]. This, combined with the option of developing hybrid versions of PCTBagging, opens limitless possibilities.

Finally, for this analysis, a version of PCTBagging was implemented, in which a completely consolidated tree was initially developed and then some nodes were eliminated to leave the desired percentage of internal nodes as consolidated nodes. However, according to the definition of PCTBagging, this is not necessary in practice. The user can decide on the specific number of nodes to be consolidated and develop the consolidated tree until this number is reached, always selecting the most populated node as the next node to be developed. Most decision tree construction algorithms include a parameter that indicates the minimum number of instances that must be in a node to allow a particular division. This parameter can be used to stop the development of a tree based on the percentage of the number of cases in the sample compared to the size of the root node. In addition, metalearning can be used to predict the size of the consolidated trees based on dataset characteristics (size, number and type of attributes, variable values, and initial entropy) to determine the number of nodes to be consolidated a priori.

CRediT authorship contribution statement

Igor Ibarguren: Investigation, Software, Writing - original draft. **Jesús M. Pérez:** Conceptualization, Methodology, Software, Formal analysis, Visualization, Writing - original draft, Writing - review & editing. **Javier Muguerza:** Conceptualization, Supervision, Funding acquisition. **Olatz Arbelaitz:** Methodology, Supervision, Funding acquisition, Writing - original draft, Writing - review & editing. **Ainhoa Yera:** Investigation, Software.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was funded by the Department of Education, Universities and Research of the Basque Government (ADIAN, IT-980-16); and by the Ministry of Economy and Competitiveness of the Spanish Government and the European Regional Development Fund - ERDF (PhysComp, TIN2017-85409-P). We would also like to thank our former undergraduate student Ander Otsoa de Alda, who participated in the implementation of the PCTBagging algorithm for the WEKA platform.

References

[1] A. Barredo Arrieta, N. Díaz-Rodríguez, J.D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Inform. Fusion 58 (2020) 82–115.

[2] P. Turney, Technical note: bias and the quantification of stability, Mach. Learn. 20 (1–2) (1995) 23–33, https://doi.org/10.1023/A:1022682001417.
[3] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[4] P. Domingos, Knowledge acquisition from examples via multiple models, in: Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), Morgan Kaufmann, 1997, pp. 98–106.
[5] J.M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, J.I. Martín, Combining multiple class distribution modified subsamples in a single tree, Pattern Recogn. Lett. 28 (4) (2007) 414–422.
[6] H. Abbasian, C. Drummond, N. Japkowicz, S. Matwin, Inner ensembles: Using ensemble methods inside the learning algorithm, in: H. Blockeel, K. Kersting, S. Nijssen, F. ++elezn++ (Eds.), Machine Learning and Knowledge Discovery in Databases, Vol. 8190 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 33–48..
[7] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann (1993).
[8] J.M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, Consolidated tree construction algorithm: Structurally steady trees, in: Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS-2004), Porto, Portugal, 2004, pp. 14–21..
[9] J.M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, A new algorithm to build consolidated trees: study of the error rate and steadiness, in: M. Kłopotek, S. Wierzchoń, K. Trojanowski (Eds.), Intelligent Information Processing and Web Mining, Advances in Soft Computing, vol. 25, Springer, Berlin Heidelberg, 2004, pp. 79–88.
[10] I. Gurrutxaga, J.M. Pérez, O. Arbelaitz, J. Muguerza, J.I. Martín, A. Ansuategi, CTC: An alternative to extract explanation from bagging, in: D. Borrajo, L. Castillo, J.M. Corchado (Eds.), Current Topics in Artificial Intelligence, no. 4788 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 90–99..
[11] J.M. Pérez, I. Albisua, O. Arbelaitz, I. Gurrutxaga, J. Martín, J. Muguerza, I. Perona, Consolidated trees versus bagging when explanation is required, Computing 89 (2010) 113–145.
[12] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[13] T.G. Dietterich, Machine-learning research: four current directions, AI Magazine 18 (1997) 97–136.
[14] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001, https://doi.org/10.1109/34.58871.
[15] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Mach. Learn. 36 (1–2) (1999) 105–139.
[16] P. Bühlmann, B. Yu, Analyzing bagging, Ann. Stat. 30 (2001) 927–961.
[17] J.M. Pérez, Árboles consolidados: construcción de un árbol de clasificación basado en múltiples submuestras sin renunciar a la explicación (Ph.D. thesis), University of the Basque, Country, 2006..
[18] G.M. Weiss, F. Provost, Learning when training and data are costly: The effect of class distribution on tree induction, J. Artif. Intell. Res. 19 (2003) 315–354.
[19] I. Ibarguren, J.M. Pérez, J. Muguerza, I. Gurrutxaga, O. Arbelaitz, Coverage based resampling: building robust consolidated decision trees, Knowl.-Based Syst. 79 (2015) 51–67, https://doi.org/10.1016/j.knosys.2014.12.023.
[20] I. Ibarguren, J.M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, An update of the J48Consolidated WEKA's class: CTC algorithm enhanced with the notion of coverage, Technical Report EHU-KAT-IK-02-14, University of the Basque Country (UPV/EHU), 2014, pp. 1–48..
[21] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.
[22] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, F. Herrera, Genetics-based machine learning for rule induction: State of the art, taxonomy and comparative study, IEEE Trans. Evol. Comput. 14 (6) (2010) 913–941.
[23] N. Japkowicz, Learning from imbalanced data sets: a comparison of various strategies, Tech. Rep. AAAI Technical Report WS-00-05, AAAI (2000).
[24] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (1) (2002) 321–357, https://doi.org/10.1613/jair.953.
[25] T. Fawcett, ROC graphs: Notes and practical considerations for researchers, Mach. Learn. 31 (2004) 1–38.
[26] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, A. Folleco, An empirical study of the classification performance of learners on imbalanced and noisy software quality data, Inf. Sci. 259 (2014) 571–595.
[27] J. Huang, C.X. Ling, Using AUC and accuracy in evaluating learning algorithms, IEEE Trans. Knowl. Data Eng. 17 (3) (2005) 299–310.
[28] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[29] I. Ibarguren, A. Lasarguren, J.M. Pérez, J. Muguerza, I. Gurrutxaga, O. Arbelaitz, BFPART: best-First PART, Inf. Sci. 367–368 (2016) 927–952.
[30] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[31] S. García, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[32] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.
[33] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18.
[34] G. Corani, A. Benavoli, A bayesian approach for comparing cross-validated algorithms on multiple data sets, Mach. Learn. 100 (2) (2015) 285–304.
[35] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks?, J Mach. Learn. Res. 17 (5) (2016) 1–10.
[36] G. Corani, A. Benavoli, J. Demšar, F. Mangili, M. Zaffalon, Statistical comparison of classifiers through bayesian hierarchical modelling, Mach. Learn. 106 (11) (2017) 1817–1837.
[37] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, J. Mach. Learn. Res. 18 (77) (2017) 1–36.
[38] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (7) (1998) 1895–1923.
[39] J. Carrasco, S. García, M. del Mar Rueda, F. Herrera, rNPBST: an R package covering non-parametric and bayesian statistical tests, in: F.J. Martínez de Pisón, R. Urraca, H. Quintián, E. Corchado (Eds.), Hybrid Artificial Intelligent Systems, Springer International Publishing, 2017, pp. 281–292..
[40] J. Carrasco, S. García, M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review, Swarm Evolut. Comput. 54 (2020) 100665.
[41] B. Calvo, G. Santafé Rodrigo, scmamp: statistical comparison of multiple algorithms in multiple problems, R J., vol. 8/1, Aug. 2016 (2016)..
[42] P.L. Bühlmann, Bagging, subagging and bragging for improving some prediction algorithms, in: Research report/Seminar für Statistik, Eidgenössische Technische Hochschule (ETH), Vol. 113, Seminar für Statistik, Eidgenössische Technische Hochschule (ETH), Zürich, 2003, pp. 1–17..
[43] G.M.-M. noz, A. Suárez, Out-of-bag estimation of the optimal sample size in bagging, Pattern Recogn. 43 (2010) 143–152..
[44] N. Landwehr, M. Hall, E. Frank, Logistic model trees, Mach. Learn. 59 (1–2) (2005) 161–205.
[45] R. Kohavi, Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 202–207.
[46] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning, 1996, pp. 148–156.
[47] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[48] R.E. Banfield, L.O. Hall, K.W. Bowyer, W. Kegelmeyer, A comparison of decision tree ensemble creation techniques, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2007) 173–180.
[49] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, Syst., Man, Cybern., Part C: Appl. Rev., IEEE Trans. 42 (4) (2012) 463–484.