On the Induction of Intelligible Ensembles

Bernhard Pfahringer

Austrian Research Institute for AI, Vienna, Austria

Abstract. Ensembles of classifiers, e.g. decision trees, often exhibit greater predictive accuracy than single classifiers alone. *Bagging* and *boosting* are two standard ways of generating and combining multiple classifiers. Unfortunately, the increase in predictive performance is usually linked to a dramatic decrease in intelligibility: ensembles are more or less black boxes comparable to neural networks. So far attempts at pruning of ensembles have not been very successful, approximately reducing ensembles into half. This paper describes a different approach which both tries to keep ensemble-sizes small during induction already and also limits the complexity of single classifiers rigorously. Single classifiers are decision-stumps of a prespecified maximal depth. They are combined by majority voting. Ensembles are induced and pruned by a simple hill-climbing procedure. These ensembles can reasonably be transformed into equivalent decision trees. We conduct some empirical evaluation to investigate both predictive accuracies and classifier complexities.

1 Introduction

Ensembles of classifiers, e.g. decision trees, often exhibit greater predictive accuracy than single classifiers alone. *Boosting* [Freund & Schapire 96] and *bagging* [Breiman 96a] are two standard ways of generating and combining multiple classifiers. Unfortunately, the increase in predictive performance is usually linked to a dramatic decrease in intelligibility: ensembles are more or less black boxes comparable to neural networks when it comes to explaining the rationale of some classificatory decision. Breiman [Breiman 96a] wrote that "What one loses, with the [bagging of] trees, is a simple and interpretable structure. What one gains is increased accuracy."

So far few have attempted to remedy this situation. Various heuristics for pruning of ensembles are introduced in [Margineantu D.D & Dietterich T.G.]. Their net result is a two-fold reduction in ensemble size at an almost identical predictive error rate. Obviously, this does not improve intelligibility significantly, as it is only marginally easier to understand the combined behaviour of just 50 decision trees instead of 100 trees. Alternatively, Kohavi & Kunz [Kohavi & Kunz 97] have investigated option decision trees which they show to be comparable in terms of predictive accuracy to bagging, but which they claim to be easier to understand and to interpret because "option decision trees provide the human with a single structure that is easier to interpret, albeit possibly very large". We somewhat doubt that option trees are really that much easier to understand, because firstly, they tend to grow quite large in practice, but more

importantly, the global effect of voting of the nested option nodes can be rather intricate and hard to grasp. This effect is similar to that of nested m-of-n tests.

This paper describes a different approach which tries to both keep ensemble-sizes small already during induction and also rigorously limits the complexity of single classifiers: only decision-stumps of a prespecified maximal depth are employed. These stumps are combined by a simple majority vote. The major advantage of small ensembles of majority-voted decision-stumps is their simple visualization. One can mechanically construct an equivalent (complex) decision tree using the precomputed majority-votes as classifications for the leaves of the tree. Furthermore, as these are a very regular decision-trees due to the way it is constructed, there are possibilities for further simplifications including logical pruning and transformation into a decision graph (In [Kohavi & Li 95] a procedure is defined for transforming so-called *oblivious* decision trees into decision graphs, and the trees constructed from ensembles of decision stumps happen to be *almost* oblivious).

The next section defines the algorithms used for selecting stumps during ensemble induction and for optional post-pruning of ensembles. Section 3 explains the experimental design and reports on the results obtained. Finally, Section 4 summarizes and elaborates upon our findings, pointing out interesting directions for further research.

2 Algorithms

As we are considering all decision stumps up to a prespecified depth for possible inclusion into the final ensemble, we need a measure for judging the quality of an ensemble. This is contrary to bagging or boosting where simply all classifiers are combined, but at least in boosting the current sub-ensemble influences the induction of the next classifier. We restrict ourselves to two-class problems. In such a context it is easy to abuse the idea of [Schapire et al 97] for explaining boosting's success as a general quality measure for ensembles. We use their margin definition for single examples. Let the two classes be labeled 0 and 1, then we have:

$$margin(ex_k) = |class(ex_k) - \frac{\sum_i^N vote_i(ex_k)}{N}|$$

Note our slightly different version using absolute values which results in the margins being bounded between 0 and 1. Straightforwardly we can define ensemble quality $eq(E)$ as the sum of the squared margins over all examples:

$$eq(E) = \sum_k margin(ex_k)^2$$

Optimal ensembles would yield a sum of 0. There is one problem with this formulation: a single overfitting classifier with zero error rate on the training set would constitute a perfect ensemble. As we limit ourselves to small decision stumps, we have not encountered this situation in practice.

2.1 Growing an ensemble

As our objective function strongly resembles the objective of linear regression, it is natural to use a related search procedure. But there is one major difference: we do not even try to determine good coefficients for the single classifiers (variables in linear regression), but instead we only try to select a reasonable subset of the possibly huge number of classifiers that are at our disposal. For this end we use the hill-climbing procedure depicted in figure 1. This procedures interleaves decision stump construction and selection and thereby eases the memory requirements of the search process. Each newly generated decision stump is either added to the current ensemble, or it may replace one of the currently employed stumps, or it may simply be discarded, depending on whatever action yields the best quality estimate.

Input: m examples, maximal depth d

```
initialize E := {};
for ds ∈ all_stumps(examples, attrs, d) do
    if size(E) < min_size then E := E + {ds}
    else
        add_q := eq(E + {ds})
        repl_q := eq(best_replacement(E, ds))
        if repl_q < min(add_q, eq(E))
            then E := best_replacement(E, ds)
            elseif add_q < min(repl_q, eq(E))
                then E := E + {ds}
return E
```

Output: an ensemble of decision stumps

Fig. 1. Growing an ensemble.

In all experiments reported in the next section we used 3 as the minimal ensemble-size and 2 as the maximal depth for decision stumps. Stumps are generated by simply constructing a complete decision-tree for the attributes selected. Splitpoints for numerical attributes are chosen by maximizing the information gain of the respective split. Given a attributes we therefore have to consider $O(a^2)$ different decision stumps.

2.2 Pruning an ensemble

As the size of ensembles generated by the growing procedure is not bounded by some upper limit, it can theoretically get quite large. Analogously to the pruning mechanism for decision trees we can define a greedy hill-climbing procedure for

pruning decision-stump ensembles as well. This simple algorithm is depicted in figure 2.

Input: ensemble E

1. **while** $eq(best\_deletion(E) < eq(E)$
2. **do** $E := best\_deletion(E)$
3. **return** E

Output: a pruned ensemble of decision stumps

Fig. 2. Pruning an ensemble.

In most experiments the effects of pruning were moderate, i.e. only a small number of stumps was usually pruned from the respective ensemble.

3 Experiments

This section reports experimental results for the algorithm described above. We compare average predictive error rates and average tree-depths for C4.5 [Quinlan 93], for ensembles of decision-stumps of max-depth 2 and for pruned ensembles on a sample of small and medium sized databases available from the UC Irvine repository [Merz & Murphy 96]. These databases are `Breast (BR)`, `Colic (CO)`, `Credit (CR)`, `Diabetes (DI)`, `German (GE)`, `Labor (LA)`, `Sonar (SO)`, `Vote (VO)`, `Lymph (L1,L2)`, `Glass (G0,..,G5)`, and `Iris (I0,I1,I2)`. Multi-class domains were transformed into N respective two-class problems in the obvious way, always trying to learn how to separate a specific class from all other classes.

To estimate predictive accuracy, stratified five-fold cross-validation was repeated five times. The rationale is this. Single cross-validation can be rather unstable [Bailey & Elkan 93], therefore one should average at least a few. Results of five-fold cross-validation do not seem to differ greatly from those of the more commonly chosen ten-fold cross-validation. But of course five-fold cross-validation is more efficient, having fewer iterations and also fewer examples in each—which makes a difference for algorithms whose complexity is non-linear.

In table 1 we have summarized the average error rates for all domains and methods. For better comparison these rates are relative with respect to the mean error rate produced by C4.5. We also plot errorbars indicating one standard deviation. It is obvious that the ensembles are able to improve on the error only in a few of the domains, but considerably so if at all. There is just a single domain were ensembles fare drastically worse than C4.5: class 5 of the `Glass` database. This is a very small class consisting of only 5% of all examples, C4.5's absolute error rate is 1.4%, whereas the error rates are 3.7% for ensembles. Thus there might be a tendency to overfit very small classes.

Real average tree depths of C4.5's trees and worst-case average tree depths for the decision-stump ensembles are summarized in table 2. This is a worst-case estimate because we simply multiply the number of stumps by their maximal depth of two. This computation does not take into account the possibilities for simplification that are exemplified further down.

There are some interesting observations to be made: ensembles can sometimes have much larger depths than C4.5's trees. Pruning of ensembles consistently reduces their depth, but not much so usually. On the other hand, even a small reduction in depth causes a huge reduction in size, as size is exponentially proportional to depth. Taking error rates and depth into account simultaneously reveals another interesting fact: when ensembles show a significantly better classification performance, their size is also significantly larger than that of C4.5's decision tree. But the inverse relationship is not necessarily true: much larger sizes may also just yield approximately equal error rates. Ensembles of decision stumps might effectively constitute a controlled way of overfitting the training set, leading to superior classification performance in those cases where C4.5 seemingly is *underfitting* the data. This will be one focus for further research.

To illustrate the claim that small ensembles of simple decision-stumps can be as easy to interpret as single decision trees, we have prepared the following series of figures. Firstly figure 3a depicts the tree induced by C4.5 for separating class 3 (labeled as 0) from all other classes (labeled as 1) in the `Glass` domain. Next, figure 3b depicts the ensemble induced for the same problem. It happens to consist of merely three stumps, two of which are effectively computing three-way discretizations of a single attribute.

Figure 4 shows the decision tree resulting from mechanically combining the three stumps of figure 3 and precomputing the voting results (class labels are 0, $\frac{1}{3}$, $\frac{2}{3}$, and 1 respectively). Obviously, this tree is quite a bit larger than the one of figure 3, but its structure is very regular. Three different kinds of *behaviour-preserving* simplifications are possible:

- Rounding of the predicted values. This will not change the categorical predictions, but we will loose information about the certainty of the respective classifications. In our example tree of figure 4 we will transform $\frac{1}{3}$ into 0, and $\frac{2}{3}$ into 1.
- Pruning of logically impossible branches, i.e. tests where previous tests further up in the tree have already constrained the possible values for the respective attribute at the node in question.
- Pruning a sub-tree where all leaves predict the same class (after rounding of class values and pruning of impossible branches).

After simplifying all these obvious redundancies the resultant tree depicted in figure 5a is comparatively simple. Nevertheless, its classificatory behaviour is necessarily *identical* to the behaviour exhibited by the decision-stump ensemble. And if so desired, this simple tree is transformable even further into a slightly simpler decision graph by transposing two tests ($A2 < 37$ and $A0 < 27$) and merging common sub-structures, as is shown in figure 5b. Admittedly, this is a

best-case example, that was wisely chosen, but in general especially post-pruned ensembles seem to stay within reasonably bounded complexity.

We have not yet automated this visualization process, but it should be straightforward to implement the mechanisms described in [Kohavi & Li 95] for this purpose.

4 Conclusions and Discussion

We have devised a simple ensemble-generating procedure which enjoys a reasonable degree of intelligibility at the expense of less dramatic improvement of predictive accuracy (when compared to e.g. boosting). There are numerous directions for further research:

- Right now the proposed mechanism is restricted to two-class problems. It should be possible to extend it to handle multi-class problems as well in way similar to how boosting can be extended for multi-class problems.
- The current mechanism faces a serious weakness in the handling of numerical attributes: as always all examples are used to construct a decision stump, the numerical cut-points are not as variable as when examples are reweighted like in boosting. Boosting is able to produce more diverse decision stumps in principle. This ability might be essential in some domains.
- It should be interesting to investigate a direct combination of boosting with the mechanism introduced in this paper.
- The current mechanism automatically adjusts the size of ensembles, thereby possibly inducing too complex ensembles. One might try to *over-prune* ensembles to further simplify them at the expense of some lost amount of ensemble quality. But the only principled way of choosing a good upper bound for ensemble sizes seems to be a (costly) wrapper-like approach based on cross-validation.
- A comparison to bayesian approaches as described in [Oliver 95, Buntine 91] is clearly necessary for judging the merits of the presented ideas.

So in summary the ideas described above can be seen as first successful steps of a much larger endeavour aimed at getting back interpretable structure of ensemble-generating procedures while at least retaining some of these procedures' inherent predictive advantages.

Acknowledgements Most of the ideas reported in this paper were conceived while the author was visiting the Computer Science Department of the University of Waikato courtesy to the following grant: "Schrödingerstipendium Nr.J01269-MAT" of the Austrian "Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)". Kai Ming Ting was very influential and helpful in numerous discussions on ensemble-generating learning procedures as was Geoff Holmes' constant urge for interpretable results. The Austrian Research Institute for Artificial Intelligence is sponsored by the Austrian Federal Ministry of Science and Transport.

References

[Bailey & Elkan 93] Bailey T.L., Elkan C.: Estimating the Accuracy of Learned Concepts, in Bajcsy R.(ed.), Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, pp.895-901, 1993.

[Breiman 96a] Breiman L.: Bagging Predictors, Machine Learning, 24(2), 1996.

[Buntine 91] Buntine W.L.: Learning Classification Trees, Statistics and Computing, Vol.2:63-73, 1991.

[Freund & Schapire 96] Freund Y., Schapire R.E.: Experiments with a New Boosting Algorithm, in Saitta L.(ed.), Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, pp.148-156, 1996.

[Kohavi & Kunz 97] Kohavi R., Kunz C.: Option Decision Trees with Majority Votes, in Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann, 1997.

[Kohavi & Li 95] Kohavi R., Li C.-H.: Oblivious Decision Trees, Graphs, and Top-Down Pruning, in Mellish C.S.(ed.), Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, pp.1071-1077, 1995.

[Margineantu D.D & Dietterich T.G.] Margineantu D.D., Dietterich T.G.: Pruning Adaptive Boosting, in Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann, 1997.

[Merz & Murphy 96] Merz, C.J., Murphy, P.M.: UCI Repository of machine learning databases, University of California, Department of Information and Computer Science, Irvine, CA, 1996.

[Oliver 95] Oliver J.J.: On Pruning and Averaging Decision Trees, in Prieditis A. & Russell S.(eds.), Proceedings of the 12th International Conference on Machine Learning (ML95), Morgan Kaufmann, San Francisco, CA, 1995.

[Quinlan 93] Quinlan J.R.: C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.

[Schapire et al 97] Schapire R.E., Freund Y., Bartlett P., Lee W.S.: Boosting the Margin: A new Explanation for the Effectiveness of Voting Methods, in Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann, 1997.

This article was processed using the LaTeX macro package with LLNCS style

Table 1. Error rates for C4.5, Ensembles, and Pruned Ensembles.

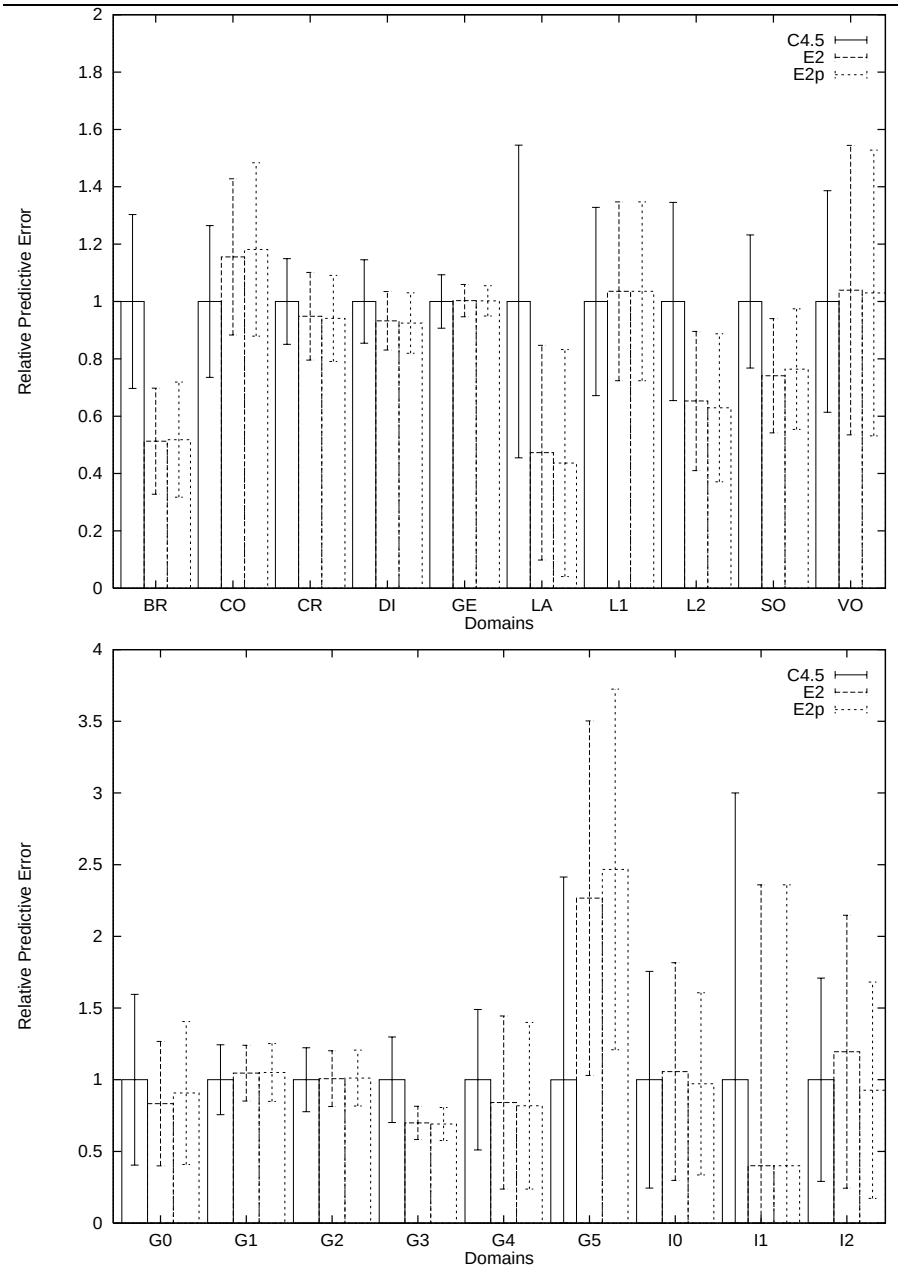
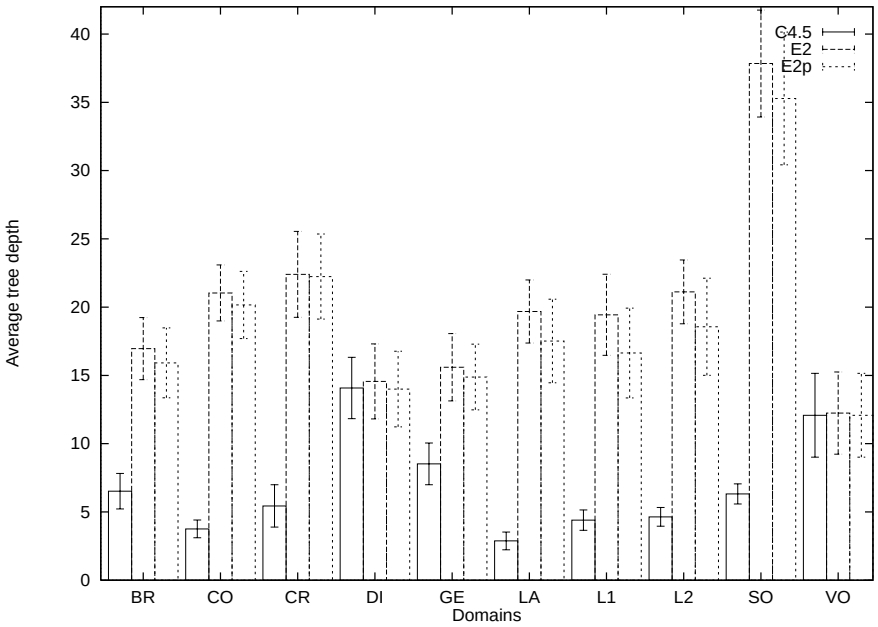


Table 2. Average tree depth for C4.5, Ensembles, and Pruned Ensembles.



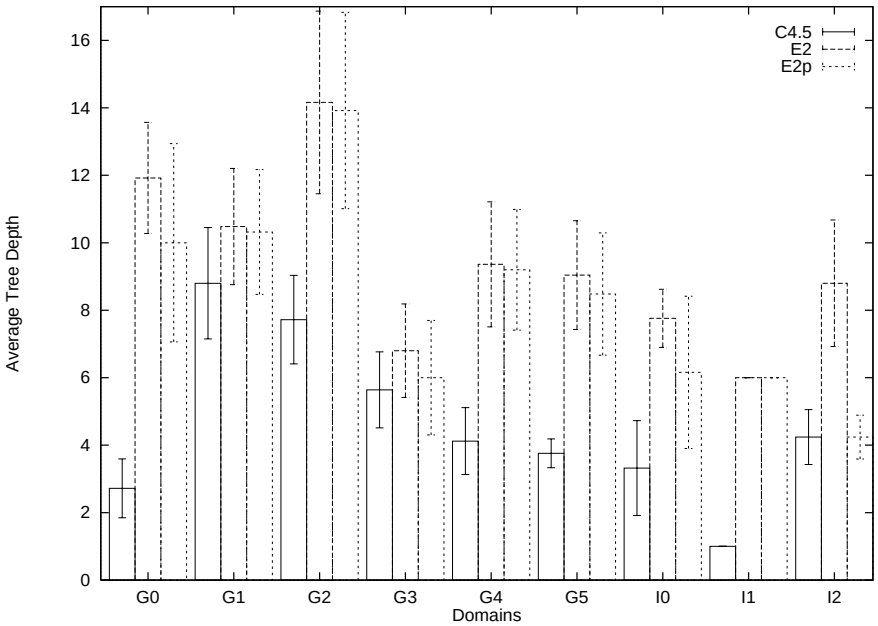


Table 3. C4's decision tree (a) and a 3 stump ensemble (b).

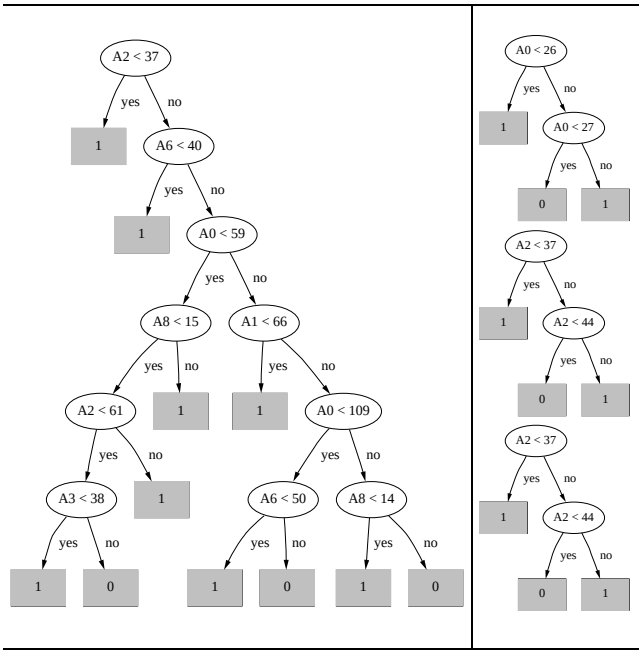


Table 4. The equivalent decision tree.

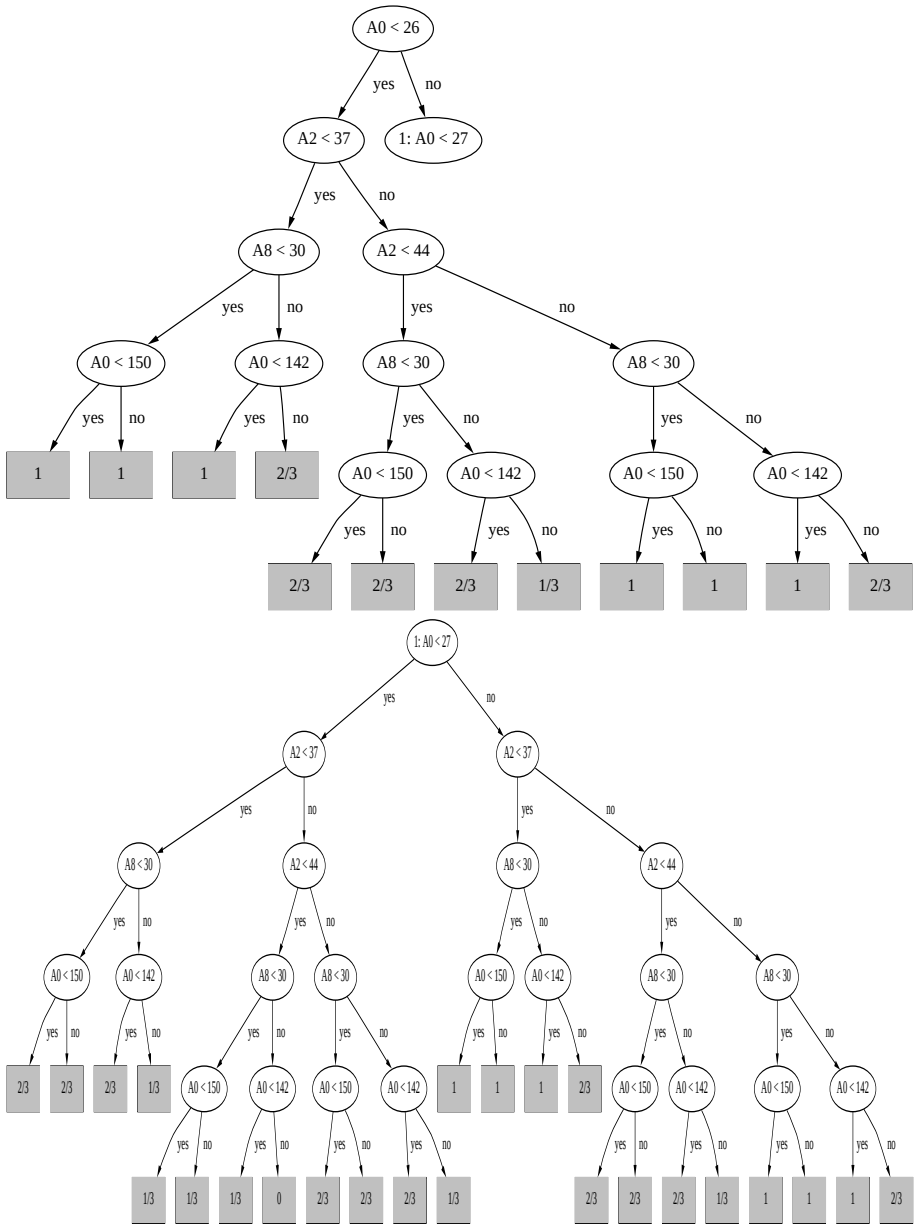


Table 5. The logically simplified decision tree (a) and graph (b).