

A genetic algorithm for interpretable model extraction from decision tree ensembles

Gilles Vandewiele, Kiani Lannoye, Olivier Janssens,
Femke Ongenaë, Filip De Turck, and Sofie Van Hoecke

Department of Information Technology
Ghent University - imec, IDLab
`{firstname}.{lastname}@UGent.be`

Abstract. Models obtained by decision tree induction techniques excel in being interpretable. However, they can be prone to overfitting, which results in a low predictive performance. Ensemble techniques provide a solution to this problem, and are hence able to achieve higher accuracies. However, this comes at a cost of losing the excellent interpretability of the resulting model, making ensemble techniques impractical in applications where decision support, instead of decision making, is crucial.

To bridge this gap, we present the GENESIM algorithm that transforms an ensemble of decision trees into a single decision tree with an enhanced predictive performance while maintaining interpretability by using a genetic algorithm. We compared GENESIM to prevalent decision tree induction algorithms, ensemble techniques and a similar technique, called ISM, using twelve publicly available data sets. The results show that GENESIM achieves better predictive performance on most of these data sets compared to decision tree induction techniques & ISM. The results also show that GENESIM's predictive performance is in the same order of magnitude as the ensemble techniques. However, the resulting model of GENESIM outperforms the ensemble techniques regarding interpretability as it has a very low complexity.

Keywords: decision support, decision tree merging, genetic algorithms

1 Introduction

Decision tree induction is a white-box machine learning technique that obtains an easily interpretable model after training. For each prediction from the model, an accompanying explanation can be given. Moreover, as opposed to rule extraction algorithms, the complete structure of the model is easy to analyze as it is encoded in a decision tree.

In domains where the decisions that need to be made are critical, the emphasis of machine learning is on offering support and advice to the experts instead of making the decisions for them. As such, the interpretability and comprehensibility of the obtained models are of primal importance for the experts that need

to base their decision on them. Therefore, a white-box approach is preferred. Examples of critical domains include the medical domain (e.g. cardiology and oncology), the financial domain (e.g. claim management and risk assessment) and law enforcement.

One of the disadvantages of decision trees is that they are prone to overfit [1]. To overcome this shortcoming, ensemble techniques have been proposed. These techniques combine the results of different classifiers, leading to an improvement in the prediction performance because of three reasons [2]. First, when the amount of training data is small compared to the size of the hypothesis space, a learning algorithm can find many different hypotheses that correctly classify all the training data, while not performing well on unseen data. By averaging the results of the different hypotheses, the risk of choosing a wrong hypothesis can be reduced. Second, many learning algorithms can get stuck in local optima. By constructing different models from different starting points, the chance to find the global optimum is increased. Third, because of the finite size of the training data set, the optimal hypothesis can be outside of the space searched by the learning algorithm. By combining classifiers, the search space gets extended, again increasing the chance to find the optimal classifier. Nevertheless, ensemble techniques also have disadvantages. First, they take considerably longer to train and make a prediction. Second, their resulting models require more storage. The third and most important disadvantage is that the obtained model consists either out of many decision trees or only one decision tree that contains uninterpretable nodes (which is the case for stacking), making it infeasible or impossible for experts to interpret and comprehend the obtained model. To bridge the gap between decision tree induction algorithms and ensemble techniques, post-processing methods are required that can convert the ensemble into a single model. By first constructing an ensemble from the data and then applying this post-processing method, a better predictive performance can possibly be achieved compared to constructing a decision tree from the data directly.

This post-processing technique is not only useful to increase the predictive performance while maintaining excellent interpretability. It can also be used in a big data setting where an interpretable model is required and the size of the training data set is too large to construct a predictive model on a single node in a feasible amount of time. To solve this, the data set can be partitioned and a predictive model can be constructed for each of these partitions in a distributed fashion. Finally, the different models can be combined together.

In this paper, we present a novel post-processing technique for ensembles, called GENetic Extraction of a Single, Interpretable Model (GENESIM), which is able to convert the different models from the ensemble into a single, interpretable model. Since each of the models in the ensemble being merged will have an impact on the predictive performance of the final combined model, a genetic approach is applied which combines models from different subsets of an ensemble. The outline of the rest of this paper is as follows. First, in Section 2, work related to genetic decision tree evolving and decision tree merging is discussed. Then, in

Section 3, the different steps of GENESIM are described. In Section 4, a comparison regarding predictive performance and model complexity is made between GENESIM, a similar technique called ISM and prevalent ensemble & decision tree induction techniques. Finally, in Section 5, a conclusion and possible future work are presented.

2 Related work

In the work of Kargupta et al. [3], decision trees are merged by first converting them to the spectral domain using a Fourier transformation. Next, the obtained spectra of different trees are added together and the inverse Fourier transformation converts the spectrum back to a decision tree. Although promising, this method has not yet been applied successfully in any real-life application.

J.R. Quinlan proposed MiniBoosting [4], wherein three boosting iterations are applied and the small resulting decision trees are merged into one very large tree, which can finally be pruned to enhance generalization. This technique has a higher accuracy than a single decision tree for the largest part of twenty-seven tested data sets, but a lower accuracy than the boosting implementation ADABOOST.

A more straight-forward technique is proposed by Quinlan [5] which translates the decision trees in production rules that are much easier to simplify than the trees themselves. Next, the production rules are either represented as a decision table, or transformed in a set of k -dimensional hyperplanes, and subsequently merged using algorithms such as the MIL algorithm [6] or respectively by calculating the intersection of the hyperplanes [7].

In the work of Van Assche et al. [8], a technique called Interpretable Single Model (ISM) is proposed. This technique is very similar to an induction algorithm, as it constructs a decision tree recursively top-down, by first extracting a fixed set of possible candidate tests from the trees in the ensemble. For each of these candidate tests, a split criterion is calculated by estimating the parameters using information from the ensemble instead of the training data. Then, the test with the optimal split criterion is chosen and the algorithm continues recursively until a pre-prune condition is met. Two shortcomings of this approach can be identified. First, information from all models, including the ones that will have a negative impact, are used to construct a final model. Second, because of the similarity with induction algorithms, it is possible to get stuck in the same local optimum as these algorithms.

Deng [9] introduced STEL, which converts an ensemble into an ordered rule list using the following steps. First, for each tree in the ensemble, each path from the root to a leaf is converted into a classification rule. After all rules are extracted, they are pruned and ranked to create an ordered rule list. This sorted rule set can

then be used for classification by iterating over each rule and returning the target when a matching rule is found. While a good predictive performance is reported for this technique, it is much harder to grasp an ordered rule list completely than a decision tree, as can be seen in Figure 1. Therefore, when interpretability is of primal importance, the post-processing technique, that converts the ensemble of models into a single model, should result in a decision tree.

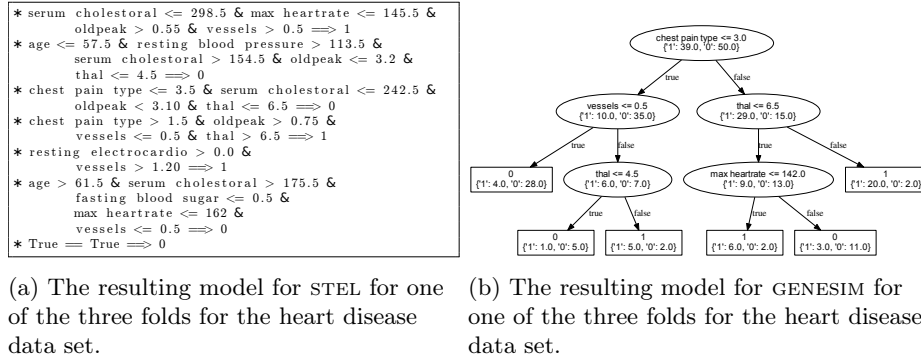


Fig. 1: Comparison of the resulting models of STEL and GENESIM regarding model complexity

It is impossible to know a priori which subset of decision trees should be merged to obtain the most accurate model. A brute-force approach that tries every possible combination would require an infeasible amount of computation time. Therefore, a genetic approach is applied that merges different decision trees for several iterations. Genetic (or evolutionary) algorithms are meta-heuristics most often used in optimization problems [10]. A recent and thorough survey of evolutionary algorithms for decision tree evolving can be found in [11].

3 GENESIM: GENetic Extraction of a Single, Interpretable Model

While in Barros et al. [11], genetic algorithms are discussed to construct decision trees from the data directly, in this paper, a genetic algorithm is applied on an ensemble of decision trees, created by using well-known induction algorithms combined with techniques such as bagging and boosting. Applying a genetic approach allows to efficiently traverse the very large search space of possible model combinations. This results in an innovative approach for merging decision trees which takes advantage of the positive properties of creating an ensemble. By exploiting multi-objective optimization, the resulting algorithm increases the accuracy and decreases the decision tree size at the same time, while most of the state-of-the-art succeeds in only one of the two.

A genetic algorithm generally consists of 6 phases, which are repeated iteratively. First, in an initialization phase, the population of candidate solutions is generated. It is important that the initial population is diverse enough, to allow for an extensive search space and reduce the chance of being stuck at local optima. Second, in each iteration, the individuals are evaluated using a fitness function. Then, in a selection phase, pairs of individuals are selected based on their fitness in order to combine them. In a fourth phase, the selected individuals are recombined, resulting in new offsprings. Furthermore, in each iteration, an individual has a certain probability to be mutated. Finally, in the end of each iteration, new offsprings are added to the population and the least fit individuals are discarded. In the subsequent subsections, each of the genetic algorithm phases are elaborated, and discussed in context of GENESIM¹.

3.1 Initialization phase

First, the training data is divided into a new training set and a validation set. Then, different induction algorithms, including C4.5, CART, QUEST and GUIDE are applied on the training data in combination with bagging. Moreover, an ADABOOST classifier is trained and each of the decision trees of its resulting model is added to the population.

3.2 Evaluation phase

The fitness function in GENESIM is defined to be the classification rate on the validation set:

$$accuracy = \frac{1}{N} * \sum_1^N \mathbb{1}_{g(x_i)=y_i}$$

with N the length of the validation data set and $g()$ the hypothesis of the individual. When two individuals have the same accuracy, the one with the lowest model complexity (expressed as number of nodes in the tree) is preferred.

3.3 Selection phase

In each iteration, deterministic tournament selection is applied to select the individuals which will get recombined in the next phase. Tournament selection has two hyper-parameters: k and p . It chooses k individuals from the population at random and sorts them by their fitness. Then, the best individual from the tournament is returned with probability p , the second best individual with probability $p * (1 - p)$, the third best with probability $p * (1 - p)^2$, and so on. In deterministic tournament selection, p is equal to 1 and thus the best individual from the tournament is always returned.

¹ <https://github.com/IBCNServices/GENESIM>

3.4 Recombination phase

To merge decision trees together, they are first converted to sets of k -dimensional hyperplanes (called the decision space), k being the number of features, by defining a unidirectional one-to-one mapping. Each node in a decision tree corresponds to a hyperplane in the decision space. Consequently, each leaf of the decision tree corresponds to a hyperrectangle in the decision space. An example of such a conversion can be seen in Figure 2.

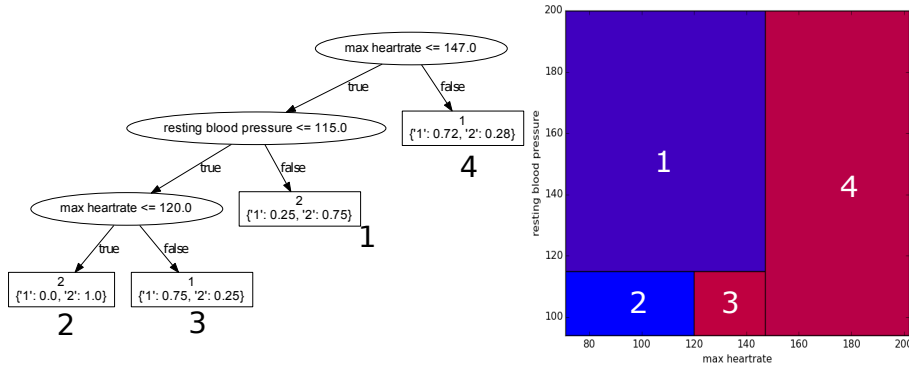


Fig. 2: Converting a decision tree to its set of k -dimensional hyperplanes. The decision tree is generated using C4.5, on the heart disease data set with two features: maximum heart rate and resting blood pressure. The color red in the decision space corresponds to class 1, the color blue corresponds to class 2. The purple tint, which consists out of a certain percentage of blue and red color, corresponds to the distribution of the two classes in a leaf.

When all the nodes from all the trees are converted to their corresponding hyperplane, the different decision spaces can be merged together by calculating their intersection using a sweep line approach discussed in [7]. In this approach, each hyperplane is projected on a line segment in each dimension. These line segments are then sorted, making it easy to find the intersecting line segments in one specific dimension. In the end, if the projected line segments of two hyperplanes intersect in each dimension, the hyperplanes intersect as well. Subsequently, their intersection can be calculated and added to the resulting decision space. This method requires $O(k * n * \log(n))$ computational time, with k the dimensionality of the data and n the number of planes in the sets, opposed to the quadratic complexity of a naive approach which calculates the intersection of each possible pair of planes.

The resulting decision spaces can contain many different regions as the number of regions in a merged space can increase quadratically in worst-case with the amount of regions in the original spaces. In order to reduce the amount of

regions in the resulting space, and thus the amount of nodes in the merged decision tree (possibly leading to better generalization), the decision space should be pruned. Pruning can be achieved by combining two regions with similar class distributions (i.e. color in Figure 2) that are next to each other. Similarity of class distributions can be measured by using a distance metric such as the Euclidean distance and subsequently comparing it with a threshold or by applying similarity metrics. It is important to note that all regions are hyperrectangles, thus the combined region should be a hyperrectangle as well. In other words, the lower and upper bound for all dimensions should be equal for both regions, except for one dimension where the lower bound in that dimension of one region is equal to the upper bound in the same dimension of the other region. For example, two candidate regions in Figure 2 are the regions 2 and 3 (but they differ too much in their class distribution to be merged).

Finally, we need to convert our merged decision space back to a decision tree. Unfortunately, the one-to-one mapping from tree to space is not bidirectional, as it is not possible to convert the set of k -dimensional hyperplanes, after the merge operation, to a uniquely defined decision tree. To solve this shortcoming, a heuristic approach is taken which identifies candidate splitting planes to create a node from, and then picks one from these candidates. To select a candidate, a metric (such as information gain) could be used, but this would introduce a bias. Therefore, a candidate is selected randomly. The candidate hyperplanes need to fulfill the constraint that they have no boundaries in all dimensions (or bounds equal to the lower and upper bound of the range of each dimension) except for one. To illustrate this, only one line can be identified as candidate line for the root node in the decision space in Figure 2. This line is unbounded in the dimension of resting blood pressure but with a value of 147 as maximum heart rate (the line left of region 4).

3.5 Mutation phase

In each iteration, an individual has a certain probability to be mutated. This can be seen as an ‘exploration’ parameter to escape local minima. Two mutation operations are defined in GENESIM: either the threshold value of a random node in the decision tree is replaced with another value or two random subtrees are swapped.

3.6 Replacement phase

The population for the next iteration is created by sorting the individuals by their fitness and only selecting the first *population_size* individuals.

4 Evaluation & results

The proposed algorithm GENESIM is compared, regarding the predictive performance and model complexity, to two ensemble methods (Random Forests (RF) [12]

& eXtreme Gradient Boosting (XGB) [13]) and four decision tree induction algorithms (C4.5 [14], CART [15], GUIDE [16] and QUEST [17]). Moreover, GENESIM is compared to ISM, which we extended with cost-complexity pruning [15]. For this, twelve data sets, having very distinct properties, from the UCI Machine Learning Repository [18] were used. An overview of the characteristics of each data set can be found in Table 1.

name	#samples	#cont	#disc	class distribution
iris	150	4	0	33.3 - 33.3 - 33.3
austra	690	5	9	55.5 - 44.5
cars	1727	0	6	70.0 - 22.2 - 4.0 - 3.8
ecoli	326	5	2	43.6 - 23.6 - 16.0 - 10.7 - 6.1
glass	213	9	0	32.4 - 35.7 - 8.0 - 6.1 - 4.2 - 13.6
heart	269	5	8	55.8 - 44.2
led7	2563	0	7	12.7 - 13.0 - 12.4 - 10.5 - 13.1 - 13.1 - 13.3 - 11.9
lymph	142	0	18	57.0 - 43.0
pima	768	7	1	65.1 - 34.9
vehicle	846	14	4	25.1 - 25.7 - 25.8 - 23.5
wine	177	13	0	32.8 - 40.1 - 27.1
wisconsinBreast	698	0	9	65.5 - 34.5

Table 1: Table with the characteristics for each data set. (#cont = number of continuous features, #disc = number of discrete features)

When the number of possible combinations was not too high, the hyper-parameters of the decision tree induction and ensemble techniques were tuned using a Grid Search technique, else Bayesian optimization was used. Unfortunately, because of a rather high complexity of GENESIM and ISM, hyper-parameter optimization could not be applied to these techniques, giving a performance advantage to the other techniques. The ensemble that was transformed into a single model by GENESIM was constructed using different induction algorithms (C4.5, CART, QUEST and GUIDE) combined with bagging and boosting. We applied 3-fold cross validation 10 times on each of the data sets and stored the mean accuracy and model complexity for the 3 folds. The mean accuracy and mean model complexity (and their corresponding standard deviations) over these 10 measurements can be found in Table 2 and Table 3. In the latter table, the average number of nodes (including the leaves) for the produced decision trees is depicted for each of the decision tree induction algorithms. For the ensemble techniques, the average number of decision trees in the constructed ensemble is depicted. Bootstrap statistical significance testing was applied to construct a Win-Tie-Loss matrix, which can be seen in Figure 3. Algorithm A wins over B for a certain data set when the mean accuracy is higher than B on that data set and the ρ -value for the bootstrap test is lower than 0.05. When an algorithm has more wins than losses compared to another algorithm, the cell is colored green (and shaded using stripes). Else, the cell is colored red (and shaded using dots). The darker the green, the more wins the algorithm has over the other. Similarly, the darker the red, the more losses an algorithm has over the other.

	XGB	CART	QUEST	GENESIM	RF	ISM	C4.5	GUIDE
heart	0.8257 $\pm 0.01\sigma$	0.7441 $\pm 0.02\sigma$	0.7585 $\pm 0.02\sigma$	0.7982 $\pm 0.02\sigma$	0.8129 $\pm 0.01\sigma$	0.8024 $\pm 0.02\sigma$	0.7877 $\pm 0.03\sigma$	0.7829 $\pm 0.02\sigma$
led7	0.8018 $\pm 0.0\sigma$	0.7997 $\pm 0.0\sigma$	0.7986 $\pm 0.0\sigma$	0.7926 $\pm 0.0\sigma$	0.8027 $\pm 0.0\sigma$	0.7996 $\pm 0.0\sigma$	0.8012 $\pm 0.0\sigma$	0.761 $\pm 0.01\sigma$
iris	0.9505 $\pm 0.01\sigma$	0.9504 $\pm 0.01\sigma$	0.9562 $\pm 0.0\sigma$	0.9463 $\pm 0.01\sigma$	0.95 $\pm 0.01\sigma$	0.9519 $\pm 0.01\sigma$	0.9395 $\pm 0.01\sigma$	0.9467 $\pm 0.01\sigma$
cars	0.9842 $\pm 0.0\sigma$	0.9749 $\pm 0.0\sigma$	0.9411 $\pm 0.01\sigma$	0.9543 $\pm 0.01\sigma$	0.9701 $\pm 0.01\sigma$	0.9685 $\pm 0.0\sigma$	0.966 $\pm 0.0\sigma$	0.9426 $\pm 0.01\sigma$
ecoli	0.8651 $\pm 0.01\sigma$	0.8196 $\pm 0.02\sigma$	0.8195 $\pm 0.01\sigma$	0.8325 $\pm 0.02\sigma$	0.8486 $\pm 0.01\sigma$	0.7507 $\pm 0.04\sigma$	0.817 $\pm 0.03\sigma$	0.8319 $\pm 0.01\sigma$
glass	0.7494 $\pm 0.02\sigma$	0.6667 $\pm 0.03\sigma$	0.649 $\pm 0.03\sigma$	0.6696 $\pm 0.03\sigma$	0.7526 $\pm 0.03\sigma$	0.6489 $\pm 0.03\sigma$	0.6763 $\pm 0.03\sigma$	0.6557 $\pm 0.02\sigma$
austra	0.8686 $\pm 0.01\sigma$	0.8506 $\pm 0.01\sigma$	0.8547 $\pm 0.01\sigma$	0.8553 $\pm 0.01\sigma$	0.8663 $\pm 0.01\sigma$	0.8557 $\pm 0.01\sigma$	0.8528 $\pm 0.01\sigma$	0.8582 $\pm 0.01\sigma$
vehicle	0.7606 $\pm 0.01\sigma$	0.6988 $\pm 0.01\sigma$	0.6986 $\pm 0.01\sigma$	0.6834 $\pm 0.01\sigma$	0.7383 $\pm 0.01\sigma$	0.6672 $\pm 0.01\sigma$	0.7115 $\pm 0.01\sigma$	0.6821 $\pm 0.01\sigma$
breast	0.9591 $\pm 0.0\sigma$	0.94 $\pm 0.01\sigma$	0.947 $\pm 0.01\sigma$	0.9496 $\pm 0.01\sigma$	0.958 $\pm 0.01\sigma$	0.9466 $\pm 0.0\sigma$	0.9443 $\pm 0.0\sigma$	0.937 $\pm 0.01\sigma$
lymph	0.8354 $\pm 0.02\sigma$	0.7686 $\pm 0.02\sigma$	0.7907 $\pm 0.03\sigma$	0.7866 $\pm 0.02\sigma$	0.817 $\pm 0.02\sigma$	0.7822 $\pm 0.03\sigma$	0.7839 $\pm 0.03\sigma$	0.7659 $\pm 0.04\sigma$
pima	0.7543 $\pm 0.01\sigma$	0.7174 $\pm 0.02\sigma$	0.7385 $\pm 0.01\sigma$	0.7266 $\pm 0.01\sigma$	0.7626 $\pm 0.01\sigma$	0.7346 $\pm 0.01\sigma$	0.7348 $\pm 0.01\sigma$	0.7285 $\pm 0.02\sigma$
wine	0.9709 $\pm 0.01\sigma$	0.9072 $\pm 0.01\sigma$	0.9055 $\pm 0.03\sigma$	0.9128 $\pm 0.03\sigma$	0.9603 $\pm 0.01\sigma$	0.8838 $\pm 0.01\sigma$	0.9217 $\pm 0.01\sigma$	0.8828 $\pm 0.03\sigma$

Table 2: Mean accuracies for the different data sets and algorithms using 10 measurements

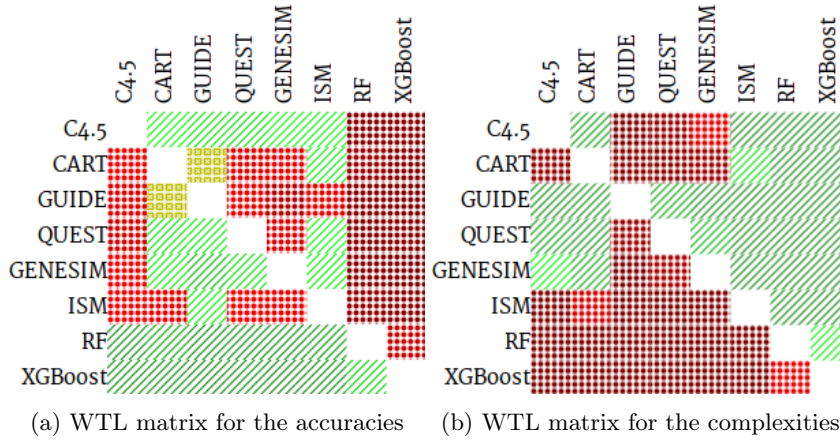


Fig. 3: Win-Tie-Loss matrices for the different algorithms for accuracies and model complexities

	XGB(*)	CART	QUEST	GENESIM	RF(*)	ISM	C4.5	GUIDE
heart	408.4815 $\pm 188.19\sigma$	35.8148 $\pm 12.54\sigma$	9.1852 $\pm 2.97\sigma$	17.4444 $\pm 4.84\sigma$	448.6113 $\pm 154.6\sigma$	35.8889 $\pm 10.71\sigma$	23.5556 $\pm 6.62\sigma$	9.1481 $\pm 2.28\sigma$
led7	459.9792 $\pm 152.17\sigma$	201.9583 $\pm 1.2\sigma$	57.625 $\pm 4.91\sigma$	92.0417 $\pm 17.08\sigma$	516.25 $\pm 155.4\sigma$	111.2917 $\pm 15.45\sigma$	58.9583 $\pm 2.09\sigma$	32.9167 $\pm 2.55\sigma$
iris	544.5238 $\pm 144.62\sigma$	12.2857 $\pm 1.34\sigma$	5.8571 $\pm 0.59\sigma$	5.9048 $\pm 0.65\sigma$	453.2381 $\pm 204.4\sigma$	10.5714 $\pm 1.91\sigma$	7.3809 $\pm 1.06\sigma$	5.3333 $\pm 0.55\sigma$
cars	631.2821 $\pm 123.71\sigma$	140.1282 $\pm 2.66\sigma$	45.6667 $\pm 4.7\sigma$	103.1539 $\pm 14.42\sigma$	438.4615 $\pm 178.3\sigma$	131.4102 $\pm 9.62\sigma$	98.4359 $\pm 4.6\sigma$	43.6154 $\pm 5.07\sigma$
ecoli	487.5625 $\pm 202.89\sigma$	35.6667 $\pm 11.77\sigma$	14.5833 $\pm 3.48\sigma$	19.0833 $\pm 4.27\sigma$	447.0623 $\pm 147.7\sigma$	60.125 $\pm 16.06\sigma$	19.25 $\pm 2.84\sigma$	10.0833 $\pm 1.43\sigma$
glass	530.7017 $\pm 179.2\sigma$	57.8421 $\pm 11.27\sigma$	22.4035 $\pm 5.66\sigma$	29.6667 $\pm 5.75\sigma$	486.9825 $\pm 160\sigma$	80.3684 $\pm 24.1\sigma$	36.2982 $\pm 3.09\sigma$	16.1579 $\pm 2.47\sigma$
austra	433.0392 $\pm 72.65\sigma$	7.7451 $\pm 6.19\sigma$	7.902 $\pm 3.23\sigma$	23.7843 $\pm 7.37\sigma$	396.3333 $\pm 181.5\sigma$	38.8824 $\pm 15.73\sigma$	26.7255 $\pm 6.82\sigma$	8.2941 $\pm 3.12\sigma$
vehicle	465.6667 $\pm 119.44\sigma$	177.1111 $\pm 22.26\sigma$	81.7778 $\pm 14.85\sigma$	83.2222 $\pm 9.68\sigma$	485.2778 $\pm 146.8\sigma$	345.5556 $\pm 45.92\sigma$	92.4444 $\pm 12.43\sigma$	33.2222 $\pm 8.71\sigma$
breast	563.3333 $\pm 170.63\sigma$	30.619 $\pm 7.89\sigma$	12.619 $\pm 3.73\sigma$	18.5238 $\pm 3.49\sigma$	395.5714 $\pm 161.4\sigma$	43.7619 $\pm 13.31\sigma$	19.4762 $\pm 2.38\sigma$	10.4286 $\pm 1.65\sigma$
lymph	608.4375 $\pm 140.47\sigma$	32.0417 $\pm 5.75\sigma$	13.5417 $\pm 3.14\sigma$	14.8333 $\pm 4.0\sigma$	497.9375 $\pm 162.3\sigma$	30.9583 $\pm 6.6\sigma$	16.9583 $\pm 2.44\sigma$	8.875 $\pm 2.81\sigma$
pima	180.0556 $\pm 85.5\sigma$	52.4445 $\pm 19.8\sigma$	12.0 $\pm 4.32\sigma$	45.2222 $\pm 8.53\sigma$	434.8334 $\pm 68.04\sigma$	101.6667 $\pm 18.5\sigma$	26.0 $\pm 5.12\sigma$	8.1111 $\pm 2.36\sigma$
wine	487.0948 $\pm 176.94\sigma$	13.4762 $\pm 1.58\sigma$	9.1905 $\pm 1.66\sigma$	8.0476 $\pm 0.93\sigma$	409.2381 $\pm 116.1\sigma$	33.3809 $\pm 3.04\sigma$	9.381 $\pm 0.33\sigma$	6.8095 $\pm 0.77\sigma$

Table 3: Mean model complexities, expressed as either number of nodes in the resulting decision tree or number of decision trees in the ensemble (*), for the different data sets and algorithms using 10 measurements

A few things can be deduced from these matrices and tables. First, we can clearly see that the ensemble techniques RF and XGB have a superior accuracy compared to all other algorithms on these data sets, and that XGB outperforms RF. While the accuracy is indeed better, the increase can be of a rather moderate size (as can be seen in Table 2). However, the resulting model is completely uninterpretable. Second, in terms of accuracy, the proposed GENESIM outperforms all decision tree induction algorithms, except C4.5. Although, GENESIM is very competitive to it. It wins on two data sets while losing on three and has no optimized hyper-parameters, in contrast to C4.5. For each data set, GENESIM used the same hyper-parameters. (such as a limited, fixed amount of iterations and using 50% of the training data as validation data). As can be seen in Figure 4, running GENESIM for a higher number of iterations could result in a better model. Third, the performance of ISM, which we extended with a post-pruning phase, is rather disappointing. Only GUIDE has a worse classification performance. Moreover, the complexity of the resulting model is higher than the other algorithms as well. Finally, GENESIM produces very interpretable models with a

very low model complexity (expressed here as the number of nodes in the tree). The average number of nodes in the resulting tree is lower than in CART and C4.5, but higher than QUEST and GUIDE. But the predictive performance of the two last-mentioned algorithms is much lower than GENESIM.

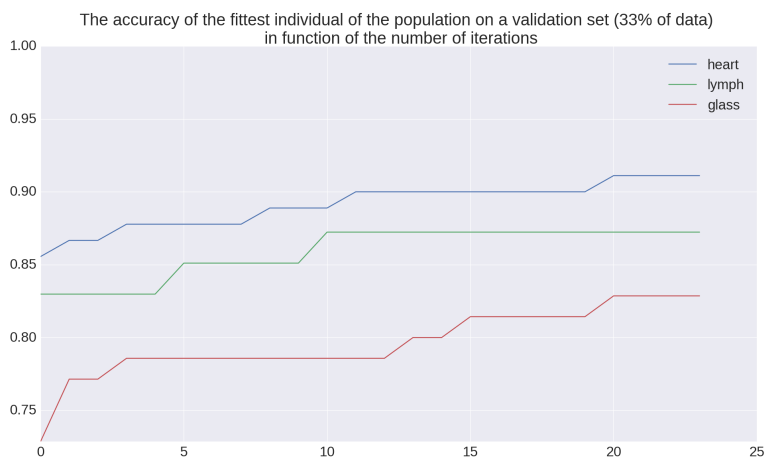


Fig. 4: The fitness (classification rate on a validation set) of the fittest population in function of the number of iterations of GENESIM

5 Conclusion

In this paper, GENESIM is proposed, a genetic approach for exploiting the positive properties of ensembles while keeping the result a single, interpretable model. GENESIM is ideally suited to support the decision-making process of experts in critical domains. Results show that in most cases, an increased predictive performance compared to naive induction algorithms can be achieved, while keeping a very similar model complexity. Results of GENESIM can still be improved by reducing the computational complexity of our algorithm, allowing hyper-parameter optimization and enabling our technique to run for more iterations in a feasible amount of time.

References

1. Donna K Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nature genetics*, 32:502–508, 2002.
2. Thomas G. Dietterich. *Multiple Classifier Systems: First International Workshop*, chapter Ensemble Methods in Machine Learning, pages 1–15. Springer Berlin Heidelberg, 2000.
3. Hillol Kargupta and Byung Hoon Park. A Fourier Spectrum-Based Approach to Represent Decision Trees for Mining Data Streams in Mobile Environments. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):216–229, 2004.
4. J. R. Quinlan. Miniboosting decision trees. *Journal of Artificial Intelligence Research*, pages 1–15, 1998.
5. J. R. Quinlan. Generating production rules from decision trees. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1*, pages 304–307. Morgan Kaufmann Publishers Inc., 1987.
6. Graham John Williams. *Inducing and combining decision structures for expert systems*. Australian National University, 1991.
7. Artur Andrzejak, Felix Langner, and Silvestre Zabala. Interpretable models from distributed data via merging of decision trees. *Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, pages 1–9, 2013.
8. Anneleen Van Assche and Hendrik Blockeel. Seeing the forest through the trees. In *International Conference on Inductive Logic Programming*, pages 269–279. Springer, 2007.
9. Houtao Deng. Interpreting tree ensembles with intrees. *arXiv preprint arXiv:1408.5456*, 2014.
10. J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
11. Rodrigo Coelho Barros, Marcio Porto Basgalupp, Andre C P L F De Carvalho, and Alex a. Freitas. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 42(3):291–312, 2012.
12. Leo Breiman. Random Forests. *Machine Learning*, 45(5):1–35, 1999.
13. Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *arXiv preprint arXiv:1603.02754*, 2016.
14. J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.
15. Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.
16. Wei-Yin Loh. Improving the precision of classification trees. *The Annals of Applied Statistics*, pages 1710–1737, 2009.
17. Wei-Yin Loh and Yu-Shan Shih. Split Selection Methods for Classification Trees. *Statistica Sinica*, 7(4):815–840, 1997.
18. M. Lichman. UCI machine learning repository, 2013.