Supplementary Material: Born-again Tree Ensembles

1. Proofs

Proof of Theorem 1. We show the NP-hardness of the born-again tree ensemble problem by reduction from 3-SAT. Let P be a propositional logic formula presented in conjunctive normal form with three literals per clause. For example, consider $\mathrm{P} = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$. 3-SAT aims to determine whether there exist literal values $x_i \in \{\textsc{True}, \textsc{False}\}$ in such a way that P is true. From a 3-SAT instance with k clauses and l literals, we construct an instance of the born-again tree ensemble problem with $2k-1$ trees of equal weight as follows:

- As illustrated in Figure 1, the first k trees $(t_1, \ldots, t_k)$ represent the clauses. Each of these trees is complete and has a depth of three, with eight leaves representing the possible combinations of values of the three literals. As a consequence of this construction, seven of the leaves predict class TRUE, and the last leaf predicts class FALSE.
- The last $k-1$ trees contain only a single leaf as root node predicting FALSE.

Finding the optimal born-again decision tree for this input leads to one of the two following outcomes:

- If the born-again decision tree contains only one leaf predicting class FALSE, then 3-SAT for P is FALSE.
- Otherwise 3-SAT for P is TRUE.

Indeed, in the first case, if the born-again tree only contains a single FALSE region (and since it is faithful to the behavior of the original tree ensemble) there exists no input sample for which TRUE represents the majority class for the $2k-1$ trees. As such, the first k trees cannot jointly predict TRUE for any input and 3-SAT is FALSE. In the second case, either the optimal born-again decision tree contains a single leaf (root node) of class TRUE, or it contains multiple leaves among which at least one leaf predicts TRUE (otherwise the born-again tree would not be optimal). In both situations, there exists a sample for which the majority class of the tree ensemble is TRUE and therefore for which all of the first k trees necessarily return TRUE, such that 3-SAT is TRUE. This argument holds for any objective involving the minimization of a monotonic size metric, i.e., a metric for which the size of a tree does not decrease upon addition of a node. This includes in particular, the depth, the number of leaves, and the hierarchical objectives involving these two metrics.

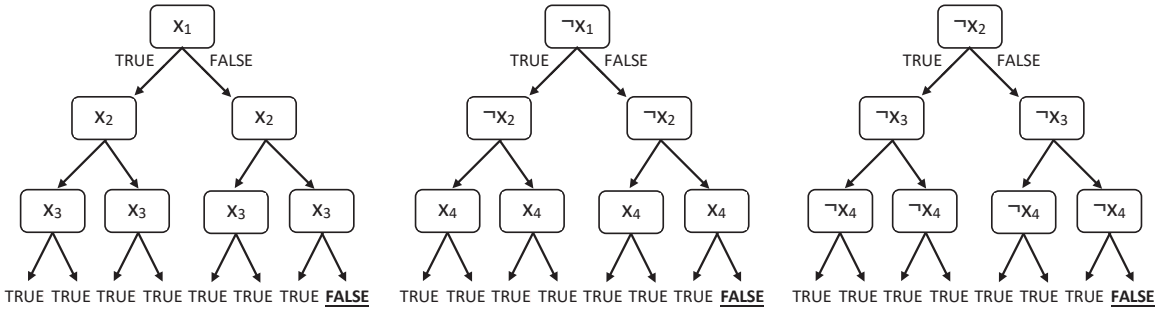

Figure 1. Trees representing the 3-SAT clauses for $\mathrm{P} = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$

Proof of Theorem 2. Consider a tree ensemble $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$. We construct a sequence of decision trees starting with $T_1 = t_1$ by iteratively appending a copy of tree t_k for $k \in \{2, \ldots, |\mathcal{T}|\}$ in place of each leaf of the tree T_{k-1} to form T_k. This leads to a born-again tree $T_{|\mathcal{T}|}$ of depth $\sum_i \Phi(t_i)$. Each leaf of this tree represents a region of the feature space over which the predicted class of the trees $t \in \mathcal{T}$ is constant, such that the ensemble behavior on this region is faithfully represented by a single class. With this construction, tree $T_{|\mathcal{T}|}$ faithfully reproduces the behavior of the original tree ensemble. Since the optimal born-again tree T has a depth no greater than that of $T_{|\mathcal{T}|}$, we conclude that $\Phi(T) \leq \sum_i \Phi(t_i)$.

Moreover, we prove that this bound is tight, i.e., it is attained for a family of tree ensembles with an arbitrary number of trees. To this end, we consider the feature space $\mathcal{X} = \mathbb{R}^d$ and the following $2d-1$ trees with equal weight:

- For $i \in \{1, \ldots, d\}$, tree t_i contains a single internal node representing the split $x_i \leq 0$, leading to a leaf node predicting class 0 when the splitting condition is satisfied, and to a leaf node predicting class 1 otherwise.
- The remaining $d - 1$ trees contain a single leaf at the root node predicting class 1.

In the resulting tree ensemble, class 0 represents the majority if and only if $x_i \leq 0$ for all $i \in \{1, \ldots, d\}$. To be faithful to the original tree ensemble, the optimal born-again decision tree must verify that $x_i \leq 0$ for all $i \in \{1, \ldots, d\}$ to declare a sample as part of class 0. This requires at least d comparisons. The depth of the born-again decision tree needed to make these tests is $\Phi(T) = d = \sum_i \Phi(t_i)$.

Proof of Theorem 3. Any tree T satisfying $F_T(\mathbf{x}) = F_{\mathcal{T}}(\mathbf{x})$ on a region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ also satisfies this condition for any subregion $(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$. Therefore, every feasible solution (tree) of the born-again tree ensemble problem for region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ is feasible for the subproblem restricted to $(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$. As a consequence, the optimal solution value $\Phi(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$ for the subproblem is smaller or equal than the optimal solution value $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ of the original problem.

Proof of Theorem 4. We will use the extended notation $\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ to denote $\mathbb{1}_{jl}$. Firstly, we observe that $\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = \mathbb{1}_{jl'}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ for all l and l'. Indeed, regardless of l and j,

$$(\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) = \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) = 0 \text{ and } F_{\mathcal{T}}(\mathbf{z}^{\text{L}}) = F_{\mathcal{T}}(\mathbf{z}^{\text{R}})) \Leftrightarrow \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = 0.$$

Next, we observe that $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'})$ and $\Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$ for all $l' > l$ follows from Theorem 3. If $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \geq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$, then $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}) \geq \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})$ follows from the two previous inequalities and:

$$\max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} = \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}) = \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\}.$$

Analogously, we observe that based on Theorem 3 $\Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})$ and $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}) \leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl})$ for all $l' < l$ holds. If $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$, then $\Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}}) \geq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'})$ follows from the two previous inequalities and:

$$\max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} = \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}}) = \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\}.$$

Combining these results with the first observation, we obtain in both cases that:

$$\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} \leq \mathbb{1}_{jl'}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\}.$$

2. Pseudo-Codes for Objectives D and DL

Our solution approach is applicable to different tree size metrics, though the binary search argument resulting from Theorem 4 is only applicable for depth minimization. We considered three possible objectives.

- (D) Depth minimization;
- (L) Minimization of the number of leaves;
- (DL) Depth minimization as primary objective, and then number of leaves as a secondary objective.

The dynamic programming algorithm for depth minimization (D) is described in the main body of the paper. Algorithms 1 and 2 detail the implementation of the dynamic programming algorithm for objectives L and DL, respectively. To maintain a similar structure and use the same solution extraction procedure in Section 3, these two algorithms focus on minimizing the number of splits rather than the number of leaves, given that these quantities are proportional and only differ by one unit in a proper binary tree. Moreover, the hierarchical objective DL is transformed into a weighted sum by associating a large cost of M for each depth increment, and 1 for each split. This allows to store each dynamic programming result as a single value and reduces memory usage.

The main differences with the algorithm for objective D occur in the loop of Line 8, which consists for L and DL in an enumeration instead of a binary search. The objective calculations are also naturally different. As seen in Line 19, the new number of splits is calculated as $1 + \Phi_1 + \Phi_2$ for objective L (i.e., the sum of the splits from the subtrees plus one). When using the hierarchical objective DL, we obtain the depth and number of splits from the subproblems as $\lfloor \Phi_i / \text{M} \rfloor$ and $\Phi_i \% \text{M}$, respectively, and use these values to obtain the new objective.

Algorithm 1 BORN-AGAIN-L($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}$)

if $(\mathbf{z}^{\text{L}} = \mathbf{z}^{\text{R}})$ **return** 0
if $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ exists in memory **then**
  return MEMORY$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$
end if
UB $\leftarrow \infty$
LB $\leftarrow 0$
for $j = 1$ **to** p and LB $<$ UB **do**
  for $l = z_j^{\text{L}}$ **to** $z_j^{\text{R}} - 1$ and LB $<$ UB **do**
    $\Phi_1 \leftarrow$ BORN-AGAIN-L$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}}))$
    $\Phi_2 \leftarrow$ BORN-AGAIN-L$(\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}), \mathbf{z}^{\text{R}})$
    if $(\Phi_1 = 0)$ and $(\Phi_2 = 0)$ **then**
      if $f(\mathbf{z}^{\text{L}}, \mathcal{T}) = f(\mathbf{z}^{\text{R}}, \mathcal{T})$ **then**
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), 0)$ **and return** 0
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), 1)$ **and return** 1
      end if
    end if

    UB $\leftarrow \min\{\text{UB}, 1 + \Phi_1 + \Phi_2\}$
    LB $\leftarrow \max\{\text{LB}, \max\{\Phi_1, \Phi_2\}\}$
  end for
end for
MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), \text{UB})$ **and return** UB

Algorithm 2 BORN-AGAIN-DL($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}$)

if $(\mathbf{z}^{\text{L}} = \mathbf{z}^{\text{R}})$ **return** 0
if $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ exists in memory **then**
  return MEMORY$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$
end if
UB $\leftarrow \infty$
LB $\leftarrow 0$
for $j = 1$ **to** p and LB $<$ UB **do**
  for $l = z_j^{\text{L}}$ **to** $z_j^{\text{R}} - 1$ and LB $<$ UB **do**
    $\Phi_1 \leftarrow$ BORN-AGAIN-DL$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}}))$
    $\Phi_2 \leftarrow$ BORN-AGAIN-DL$(\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}), \mathbf{z}^{\text{R}})$
    if $(\Phi_1 = 0)$ and $(\Phi_2 = 0)$ **then**
      if $f(\mathbf{z}^{\text{L}}, \mathcal{T}) = f(\mathbf{z}^{\text{R}}, \mathcal{T})$ **then**
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), 0)$ **and return** 0
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$,M+1) **and return** M+1
      end if
    end if
    DEPTH $\leftarrow 1 + \max\{\lfloor \Phi_1/\text{M} \rfloor, \lfloor \Phi_2/\text{M} \rfloor\}$
    SPLITS $\leftarrow 1 + \Phi_1 \% \text{M} + \Phi_2 \% \text{M}$
    UB $\leftarrow \min\{\text{UB}, \text{M} \times \text{DEPTH} + \text{SPLITS}\}$
    LB $\leftarrow \max\{\text{LB}, \max\{\Phi_1, \Phi_2\}\}$
  end for
end for
MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), \text{UB})$ **and return** UB

3. Solution Extraction

To reduce computational time and memory consumption, our dynamic programming algorithms only store the optimal objective value of the subproblems. To extract the complete solution, we exploit the following conditions to recursively retrieve the optimal splits from the DP states. For a split to belong to the optimal solution:

1. Both subproblems should exist in the dynamic programming memory.
2. The objective value calculated from the subproblems should match the known optimal value for the considered region.

These conditions lead to Algorithm 3, which reports the optimal tree in DFS order.

4. Detailed Results

In this section, we report additional computational results which did not fit in the main paper due to space limitations. Tables A1 to A3 extend the results of Tables 2 and 3 in the main paper. They report for each objective the depth and number of leaves of the different classifiers, as well as their minimum and maximum values achieved over the ten runs (one for each training/test pair).

Finally, Tables A4 to A6 extend the results of Table 4 in the main paper. They report for each objective the average accuracy and F1 scores of the different classifiers, as well as the associated standard deviations over the ten runs on different training/test pairs.

Algorithm 3 EXTRACT-OPTIMAL-SOLUTION($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}, \Phi_{\text{OPT}}$)

if $\Phi_{\text{OPT}} = 0$ **then**
  EXPORT a leaf with class MAJORITY-CLASS($\mathbf{z}^{\text{L}}$)
  return
else
  for $j = 1$ **to** p **do**
    for $l = z_j^{\text{L}}$ **to** $z_j^{\text{R}} - 1$ **do**
      $\Phi_1 \leftarrow$ MEMORY($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}})$)
      $\Phi_2 \leftarrow$ MEMORY($\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}, \mathbf{z}^{\text{R}}$)
      if $\Phi_{\text{OPT}} =$ CALCULATE-OBJECTIVE(Φ_1, Φ_2) **then**
        EXTRACT-OPTIMAL-SOLUTION($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}}), \Phi_1$)
        EXTRACT-OPTIMAL-SOLUTION($\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}), \mathbf{z}^{\text{R}}, \Phi_2$)
        EXPORT a split on feature j with level z_j^{L}
        return
      end if
    end for
  end for
end if

Table A1. Complexity of the different classifiers – Considering objective D

	Random Forest			Born Again Tree						Born Again Tree + Pruning					
	#Leaves			Depth			#Leaves			Depth			#Leaves		
Data set	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
BC	61.1	57	68	12.5	11	13	2279.4	541	4091	9.1	8	11	35.9	26	44
CP	46.7	40	55	8.9	7	11	119.9	23	347	7.0	4	9	31.2	10	50
FI	47.3	40	52	8.6	3	13	71.3	5	269	6.5	3	9	15.8	4	27
HT	42.6	36	49	6.0	2	7	20.2	3	38	5.1	2	6	13.2	3	22
PD	53.7	45	63	9.6	7	12	460.1	101	1688	9.4	7	12	79.0	53	143
SE	55.7	51	60	10.2	9	11	450.9	159	793	7.5	6	8	21.5	16	31
Overall	51.2	36	68	9.3	2	13	567.0	3	4091	7.4	2	12	32.8	3	143

Table A2. Complexity of the different classifiers – Considering objective L

	Random Forest			Born Again Tree						Born Again Tree + Pruning					
	#Leaves			Depth			#Leaves			Depth			#Leaves		
Data set	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
BC	61.1	57	68	18.0	17	20	890.1	321	1717	9.0	7	11	23.1	17	32
CP	46.7	40	55	8.9	7	11	37.1	10	105	6.5	3	8	11.4	4	21
FI	47.3	40	52	8.6	3	13	39.2	4	107	6.3	3	8	12.0	4	20
HT	42.6	36	49	6.3	2	8	11.9	3	19	4.3	2	6	6.4	3	9
PD	53.7	45	63	15.0	12	19	169.7	50	345	11.0	8	17	30.7	20	42
SE	55.7	51	60	13.8	12	16	214.6	60	361	7.7	6	9	14.2	9	19
Overall	51.2	36	68	11.8	2	20	227.1	3	1717	7.5	2	17	16.3	3	42

Table A3. Complexity of the different classifiers – Considering objective DL

	Random Forest			Born Again Tree						Born Again Tree + Pruning					
	#Leaves			Depth			#Leaves			Depth			#Leaves		
Data set	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
BC	61.1	57	68	12.5	11	13	1042.3	386	2067	8.9	8	10	27.7	18	39
CP	46.7	40	55	8.9	7	11	37.1	10	105	6.5	3	8	11.4	4	21
FI	47.3	40	52	8.6	3	13	39.2	4	107	6.3	3	8	12.0	4	20
HT	42.6	36	49	6.0	2	7	12.0	3	19	4.6	2	6	6.5	3	10
PD	53.7	45	63	9.6	7	12	206.7	70	387	8.9	7	11	42.1	28	62
SE	55.7	51	60	10.2	9	11	261.0	65	495	7.4	6	9	17.0	12	24
Overall	51.2	36	68	9.3	2	13	266.4	3	2067	7.1	2	11	19.5	3	62

Table A4. Accuracy of the different classifiers – Considering objective D

	Random Forest				Born Again Tree				Born Again Tree + Pruning			
	Acc.		F1		Acc.		F1		Acc.		F1	
Data set	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
BC	0.953	0.040	0.949	0.040	0.953	0.040	0.949	0.040	0.946	0.047	0.941	0.046
CP	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024
FI	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049
HT	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044
PD	0.746	0.062	0.692	0.065	0.746	0.062	0.692	0.065	0.750	0.067	0.700	0.069
SE	0.790	0.201	0.479	0.207	0.790	0.201	0.479	0.207	0.790	0.196	0.481	0.208
Avg.	0.804	0.064	0.728	0.072	0.804	0.064	0.728	0.072	0.803	0.065	0.729	0.073

Table A5. Accuracy of the different classifiers – Considering objective L

	Random Forest				Born Again Tree				Born Again Tree + Pruning			
	Acc.		F1		Acc.		F1		Acc.		F1	
Data set	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
BC	0.953	0.040	0.949	0.040	0.953	0.040	0.949	0.040	0.943	0.052	0.938	0.053
CP	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024
FI	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049
HT	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044
PD	0.746	0.062	0.692	0.065	0.746	0.062	0.692	0.065	0.751	0.064	0.698	0.068
SE	0.790	0.201	0.479	0.207	0.790	0.201	0.479	0.207	0.790	0.193	0.479	0.207
Avg.	0.804	0.064	0.728	0.072	0.804	0.064	0.728	0.072	0.803	0.065	0.727	0.074

Table A6. Accuracy of the different classifiers – Considering objective DL

	Random Forest				Born Again Tree				Born Again Tree + Pruning			
	Acc.		F1		Acc.		F1		Acc.		F1	
Data set	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
BC	0.953	0.040	0.949	0.040	0.953	0.040	0.949	0.040	0.941	0.051	0.935	0.049
CP	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024
FI	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049
HT	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044
PD	0.746	0.062	0.692	0.065	0.746	0.062	0.692	0.065	0.747	0.069	0.693	0.076
SE	0.790	0.201	0.479	0.207	0.790	0.201	0.479	0.207	0.781	0.195	0.477	0.210
Avg.	0.804	0.064	0.728	0.072	0.804	0.064	0.728	0.072	0.801	0.066	0.726	0.075

5. Born-Again Tree Illustration

Finally, Figure 2 illustrates the born-again tree ensemble problem on a simple example with an original ensemble composed of three trees. All cells and the corresponding majority classes are represented. There are two classes, depicted by a ● and a ○ sign, respectively.

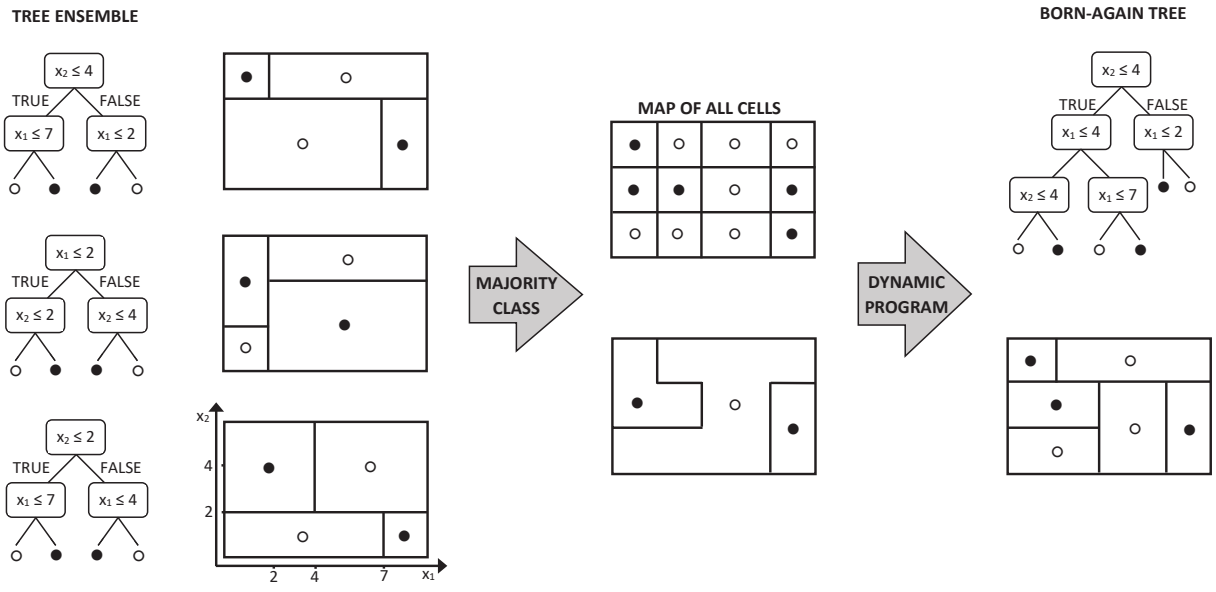

Figure 2. Cells and born-again tree on a simple example