Combinatorial Optimization and Machine Learning – Part II

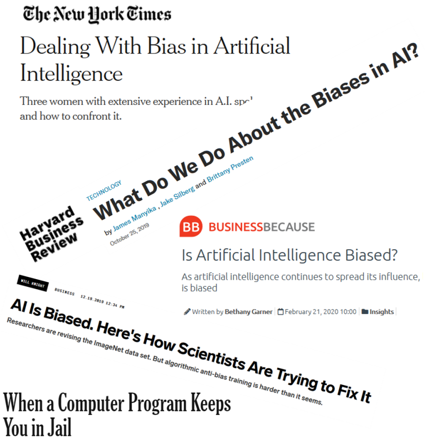
Thibaut Vidal[1], Thiago Serra[2]

[1] Departamento de Informática, PUC-Rio
vidalt@inf.puc-rio.br
[2] Freeman College of Management, Bucknell University
thiago.serra@bucknell.edu

Image Credit: Hitchhikers Guide to the Galaxy

The rise of interpretable/explainable AI

- Machine learning is becoming more and more widespread for high stakes decisions:
  - Recurrence predictions in medicine
  - Credit default risk evaluations...
  - ...and even for some applications where it should not be applied in the first place ⇒ Release decisions in criminal justice.



The rise of interpretable/explainable AI

Interpretability $\neq$ Explainability

- The decision process of an **interpretable model** can be understood **by design**, e.g., it is possible to track the individual decisions taken in a decision tree. This implies that interpretable models are also **transparent**.
- The internal process of an **explainable model** does not need to be transparent (it can be a black box), but an **additional algorithm** should be available to explain the decisions taken (e.g., saliency maps, feature relevance analysis).

The rise of interpretable/explainable AI

Interpretability $\neq$ Explainability

- Explainable AI techniques pose one issue: in case of an unexpected behavior or bias, two algorithms/models may need debugging instead of one [21].
- We will therefore principally focus on interpretable AI algorithms in this talk, considering two important classes of methods: ensembles of trees and (ReLU) neural networks.

Decision Trees and Random Forests

Decision tree:

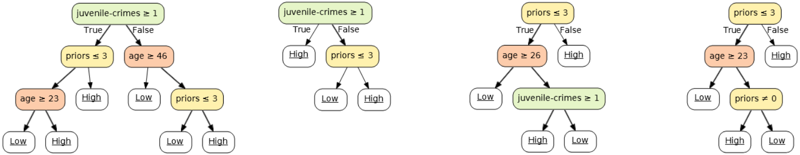
++ Simple and explainable

– – Possible overfit & typically lower accuracy on test data

Tree ensemble – Random forest:

++ Ensemble learning algorithm: **better generalization** on test data

– – **Lack of interpretability**



Related Literature

Thinning tree ensembles

Pruning some weak learners [16, 20, 23, 26]

Replacing the tree ensemble by a simpler classifier [1, 6, 17, 24]

Rule extraction via bayesian model selection [13]

Extracting a single tree from a tree ensemble by actively sampling training points [2, 3]

Thinning neural networks

Model compression and knowledge distillation [7, 14]: Using a "teacher" to train a compact "student' with similar knowledge.

Creating soft decision trees from a neural network [10], or decomposing the gradient in knowledge distillation [11].

Simplifying neural networks [8, 9, 22].

Optimal decision trees

Linear programming algorithms have been exploited to find linear combination splits [4].

Extensive study of global optimization methods, based on mixed-integer programming or dynamic programming, for the construction of optimal decision trees [5, 12, 15, 18, 25]

- Other, model-agnostic, explanation approaches such as LIME [19].
  - ⇒ Aimed at providing a **local** explanation.
  - ⇒ Works by training a simpler surrogate model (e.g., a linear classifier) around an instance that should be explained and analyze the weights.

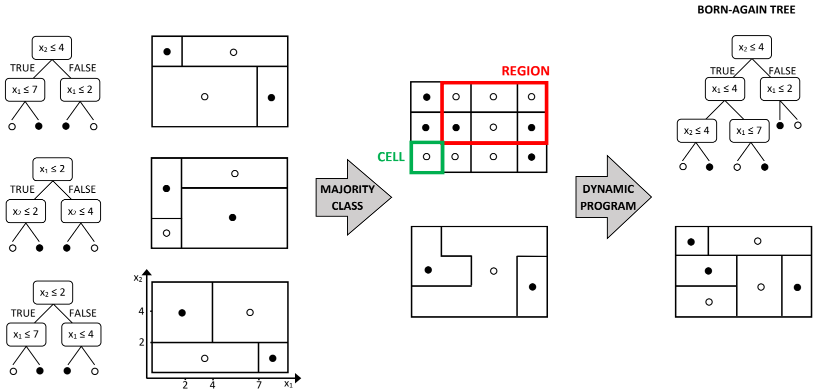
Born-Again Tree Ensembles

- A recent exact algorithm that transforms a tree ensemble into a born-again decision tree (BA tree) that is:
  - **Optimal** in size (number of leaves or depth), and
  - **Faithful** to the tree ensemble **in its entire feature space**.
- The BA tree is effectively **a different representation of the same decision function**.

A single —minimal-size— decision tree that faithfully reproduces the decision function of the random forest.

Methodology

Construction Process



Methodology

Problem 1: Born-Again Tree Ensemble

Given a tree ensemble $\mathcal{T}$, we search for a decision tree T of **minimal size** such that $F_T(\mathbf{x}) = F_{\mathcal{T}}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^p$.

Theorem 1

Problem 1 is NP-hard when optimizing depth, number of leaves, or any hierarchy of these two objectives.

Verifying that a given solution is feasible (faithful) is NP-hard.

Methodology

Dynamic Program 1

Let $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ be the depth of an optimal born-again decision tree for a region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$. Then:

$$\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = \begin{cases} 0 \quad \text{if} \quad \textsc{id}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) \\ \min\limits_{1 \leq j \leq p} \left\{ \min\limits_{z_j^{\text{L}} \leq l < z_j^{\text{R}}} \left\{ 1 + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} \right\} \right\}, \end{cases}$$

in which $\textsc{id}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ takes value TRUE iff all cells $\mathbf{z}$ such that $\mathbf{z}^{\text{L}} \leq \mathbf{z} \leq \mathbf{z}^{\text{R}}$ are from the same class (i.e. base case).

Issue 1

Detecting base cases

Issue 2

Numerous recursive calls

Circumventing Issue 1

We tried several alternatives to efficiently check base cases. The best approach we found consisted in including the base case evaluation within the DP:

Dynamic Program 2

Let $\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}})$ be the depth of an optimal born-again decision tree for a region $(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}})$. Then:

$$\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}) = \min_{1 \leq j \leq p} \left\{ \min_{z_j^{\mathrm{L}} \leq l < z_j^{\mathrm{R}}} \left\{ \mathbb{1}_{jl}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}) + \max\{\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl}), \Phi(\mathbf{z}^{\mathrm{L}}_{jl}, \mathbf{z}^{\mathrm{R}})\} \right\} \right\}$$

$$\text{where } \mathbb{1}_{jl}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}) = \begin{cases} 0 & \text{if } \quad \Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl}) = \Phi(\mathbf{z}^{\mathrm{L}}_{jl}, \mathbf{z}^{\mathrm{R}}) = 0 \\ & \text{and } \; F_{\mathcal{T}}(\mathbf{z}^{\mathrm{L}}) = F_{\mathcal{T}}(\mathbf{z}^{\mathrm{R}}); \\ 1 & \text{otherwise.} \end{cases}$$

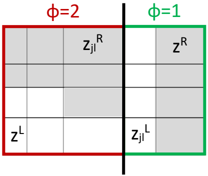
Circumventing Issue 2

We exploit two simple properties to reduce the number of recursive calls:

Property 2

If $\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl}) \geq \Phi(\mathbf{z}^{\mathrm{L}}_{jl}, \mathbf{z}^{\mathrm{R}})$ then for all $l' > l$:

$$\begin{aligned}\mathbb{1}_{jl}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}) + \max\{\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl}), \Phi(\mathbf{z}^{\mathrm{L}}_{jl}, \mathbf{z}^{\mathrm{R}})\} \\ \leq \mathbb{1}_{jl'}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}) + \max\{\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl'}), \Phi(\mathbf{z}^{\mathrm{L}}_{jl'}, \mathbf{z}^{\mathrm{R}})\}\end{aligned}$$



Property 3

If $\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl}) \leq \Phi(\mathbf{z}^{\mathrm{L}}_{jl}, \mathbf{z}^{\mathrm{R}})$ then for all $l' < l$:

$$\begin{aligned}\mathbb{1}_{jl}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}) + \max\{\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl}), \Phi(\mathbf{z}^{\mathrm{L}}_{jl}, \mathbf{z}^{\mathrm{R}})\} \\ \leq \mathbb{1}_{jl'}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}) + \max\{\Phi(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}_{jl'}), \Phi(\mathbf{z}^{\mathrm{L}}_{jl'}, \mathbf{z}^{\mathrm{R}})\}\end{aligned}$$

Allowing us to search for the best hyperplane level for each feature with a binary search.

Algorithm 1 $\textsc{Born-Again}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}})$

if $(\mathbf{z}^{\mathrm{L}} = \mathbf{z}^{\mathrm{R}})$ **return** 0
if $(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}})$ exists in memory **return** $\textsc{Memory}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}})$
$(\mathrm{LB}, \mathrm{UB}) \leftarrow (0, \infty)$
for $j = 1$ **to** p and $\mathrm{LB} < \mathrm{UB}$ **do**
  $(\textsc{Low}, \textsc{Up}) \leftarrow (z_j^{\mathrm{L}}, z_j^{\mathrm{R}})$
  while $\textsc{Low} < \textsc{Up}$ and $\mathrm{LB} < \mathrm{UB}$ **do**
    $l \leftarrow \lfloor (\textsc{Low} + \textsc{Up})/2 \rfloor$
    $\Phi_1 \leftarrow \textsc{Born-Again}(\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}} + \mathbf{e}_j(l - z_j^{\mathrm{R}}))$
    $\Phi_2 \leftarrow \textsc{Born-Again}(\mathbf{z}^{\mathrm{L}} + \mathbf{e}_j(l + 1 - z_j^{\mathrm{L}}), \mathbf{z}^{\mathrm{R}})$
    if $(\Phi_1 = 0)$ and $(\Phi_2 = 0)$ **then**
      if $f(\mathbf{z}^{\mathrm{L}}, \mathcal{T}) = f(\mathbf{z}^{\mathrm{R}}, \mathcal{T})$ **then** $\textsc{Memorize}((\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}), 0)$ **and return** 0
      else $\textsc{Memorize}((\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}), 1)$ **and return** 1
    end if
    $\mathrm{UB} \leftarrow \min\{\mathrm{UB}, 1 + \max\{\Phi_1, \Phi_2\}\}$
    $\mathrm{LB} \leftarrow \max\{\mathrm{LB}, \max\{\Phi_1, \Phi_2\}\}$
    if $(\Phi_1 \geq \Phi_2)$ **then** $\textsc{Up} \leftarrow l$
    if $(\Phi_1 \leq \Phi_2)$ **then** $\textsc{Low} \leftarrow l + 1$
  end while
end for
$\textsc{Memorize}((\mathbf{z}^{\mathrm{L}}, \mathbf{z}^{\mathrm{R}}), \mathrm{UB})$ **and return** UB

Experimental Analyses

Goals

Evaluate the scalability of the DP algorithm depending on:

- the size metric in use
- the number of trees in the ensemble
- the number of samples and features in the datasets

Study the structure and complexity of the born-again trees for different size metrics.
Measure the impact of an a-posteriori pruning strategy.

Experimental Analyses

Datasets

We used datasets from diverse applications, including medicine (BC, PD), criminal justice (COMPAS), and credit scoring (FICO).

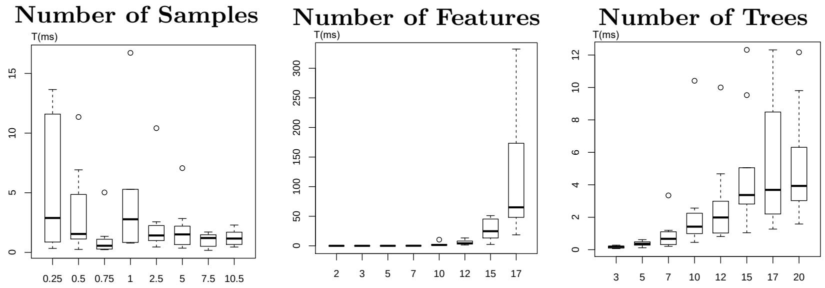
Data set	n	p	K	CD	Src.
BC – Breast-Cancer	683	9	2	65-35	UCI
CP – COMPAS	6907	12	2	54-46	HuEtAl
FI – FICO	10459	17	2	52-48	HuEtAl
HT – HTRU2	17898	8	2	91-9	UCI
PD – Pima-Diabetes	768	8	2	65-35	SmithEtAl
SE – Seeds	210	7	3	33-33-33	UCI

Data Preparation

One-hot encoding for categorical variables.
Continuous variables binned into ten ordinal scales.
Generate training and test samples for all data sets by ten-fold cross validation.
For each fold and each dataset, generate a random forest composed of 10 trees with a depth of 3.

Experimental Analyses

Scalability



Computational time(ms) of the DP as a function of the number of samples, features and trees.

Experimental Analyses

Simplicity

Depth and number of leaves of the born-again trees:

	D		L		DL	
Data set	Depth	# Leaves	Depth	# Leaves	Depth	# Leaves
BC	12.5	2279.4	18.0	890.1	12.5	1042.3
CP	8.9	119.9	8.9	37.1	8.9	37.1
FI	8.6	71.3	8.6	39.2	8.6	39.2
HT	6.0	20.2	6.3	11.9	6.0	12.0
PD	9.6	460.1	15.0	169.7	9.6	206.7
SE	10.2	450.9	13.8	214.6	10.2	261.0
Avg.	9.3	567.0	11.8	227.1	9.3	266.4

Analysis

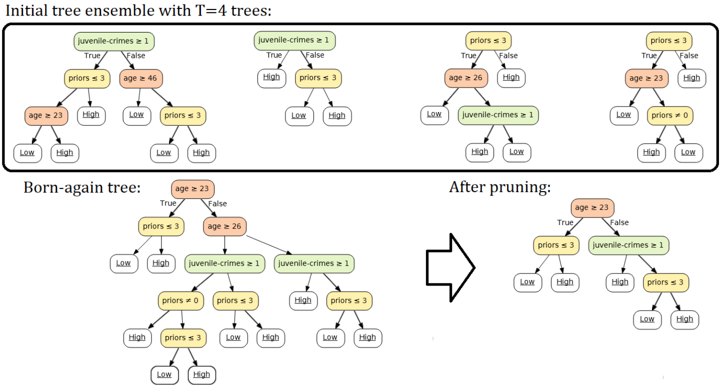
The decision function of a random forest is visibly complex
One main reason: *Incompatible feature combinations* are being represented, and the decision function of the RF is not necessarily uniform on these regions due to the other features.

Experimental Analyses

Post-Pruning

Eliminate inexpressive tree sub-regions. From bottom to top:

- Verify whether both sides of a split contain at least one sample
- Eliminate every such *empty* split



Experimental Analyses

Analysis

With post-pruning, faithfulness is no longer guaranteed per definition. We need to experimentally evaluate:

- Impact on simplicity
- Impact on accuracy

Depth and number of leaves:

	RF	BA-Tree		BA+P	
	Leaves	Depth	Leaves	Depth	Leaves
BC	61.1	12.5	2279.4	9.1	35.9
CP	46.7	8.9	119.9	7.0	31.2
FI	47.3	8.6	71.3	6.5	15.8
HT	42.6	6.0	20.2	5.1	13.2
PD	53.7	9.6	460.1	9.4	79.0
SE	55.7	10.2	450.9	7.5	21.5
Avg.	51.2	9.3	567.0	7.4	32.8

Accuracy and F1 score comparison:

	RF		BA-Tree		BA+P	
	Acc	F1	Acc	F1	Acc	F1
BC	0.953	0.949	0.953	0.949	0.946	0.941
CP	0.660	0.650	0.660	0.650	0.660	0.650
FI	0.697	0.690	0.697	0.690	0.697	0.690
HT	0.977	0.909	0.977	0.909	0.977	0.909
PD	0.746	0.692	0.746	0.692	0.750	0.700
SE	0.790	0.479	0.790	0.479	0.790	0.481
Avg.	0.804	0.728	0.804	0.728	0.803	0.729

Heuristic Solutions

The current DP approach can be applied to datasets with up to 20 features in our experiments. To solve larger cases we introduced a heuristic **that guarantees faithfulness, but relaxes optimality**.

- Instead of opening all recursions, it uses a greedy split criterion (information gain) considering $n_c = 100$ random cells within the region.
- If the n_c cells belong to the same class, it uses a resource-constrained shortest path bound to attempt to prove that all cells within this region belong to the same class.
- If this bound is insufficient, a MIP is used to prove uniformity or detect a violating cell.

This heuristic finds faithful BA-trees for large datasets (Ionosphere, Spambase, and Miniboone, the later with over 130,000 samples and 50 features) in less than 30 seconds.
The depth and number of leaves increases by 22.90% and 18.20% on average over the optimal solutions, but the heuristic solutions usually give good trade-offs.

Conclusions

- BA-trees provide a compact representations of the decision functions of random forests, as a single —minimal size— decision tree.
- Sheds a new light on **random forests visualization and interpretability**.
- Progressing towards interpretable models is an important step towards addressing bias and data mistakes in learning algorithms.
- Optimal classifiers can be fairly complex. Indeed, BA-trees reproduce the complete decision function for *all regions of the feature space*.
    - Pruning can solve this issue
    - Heuristics can be used for datasets which are too large to be solved to optimality

Paths towards algorithmic interpretability

- Trade-off: Faithfulness vs Simplicity
  - **Heuristic** BA-trees can permit to reduce computational time and apply the concept to larger data sets
  - Faithfulness can be traded in exchange for more simplicity, but this requires identifying the most "meaningful" regions of the feature space.
  - The algorithm can be focused on smaller sub-regions to provide local explanations
- Other research paths:
  - Extension to gradient boosting, regression applications etc.
  - Interpretation and simplification of other ML models (next presentation by Thiago Serra)
  - Counterfactual explanations, Feature importance measurement...

Thanks !

THANK YOU FOR YOUR ATTENTION !

Contact me:
`vidalt@inf.puc-rio.br`

Articles, slides and data sets:
`http://w1.cirrelt.ca/~vidalt/`

Source codes:
`https://github.com/vidalthi/`

Regular updates:
`https://twitter.com/vidalthi`

Bibliography I

[1] Bai, J., Y. Li, J. Li, Y. Jiang, S. Xia. 2019. Rectified decision trees: Towards interpretability, compression and empirical soundness. *arXiv preprint arXiv:1903.05965* .

[2] Bastani, O., C. Kim, H. Bastani. 2017. Interpretability via model extraction. *arXiv preprint arXiv:1706.09773* .

[3] Bastani, O., C. Kim, H. Bastani. 2017. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504* .

[4] Bennett, K. 1992. Decision tree construction via linear programming. *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, Utica, Illinois*.

[5] Bertsimas, D., J. Dunn. 2017. Optimal classification trees. *Machine Learning* **106**(7) 1039–1082.

[6] Breiman, L., N. Shang. 1996. Born again trees. Tech. rep., University of California Berkeley.

[7] Buciluǎ, C., R. Caruana, A. Niculescu-Mizil. 2006. Model compression. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[8] Clark, K., M.-T. Luong, U. Khandelwal, C. D. Manning, Q. V. Le. 2019. Bam! born-again multi-task networks for natural language understanding. *arXiv preprint arXiv:1907.04829* .

[9] Frankle, J., M. Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* .

Bibliography II

[10] Frosst, N., G. Hinton. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784* .

[11] Furlanello, Tommaso, Zachary C Lipton, Michael Tschannen, Laurent Itti, Anima Anandkumar. 2018. Born again neural networks. *arXiv preprint arXiv:1805.04770* .

[12] Günlük, O., J. Kalagnanam, M. Menickelly, K. Scheinberg. 2018. Optimal decision trees for categorical data via integer programming. *arXiv preprint arXiv:1612.03225* .

[13] Hara, S., K. Hayashi. 2016. Making tree ensembles interpretable: A bayesian model selection approach. *arXiv preprint arXiv:1606.09066* .

[14] Hinton, G., O. Vinyals, J. Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

[15] Hu, X., C. Rudin, M. Seltzer. 2019. Optimal sparse decision trees. *Advances in Neural Information Processing Systems*.

[16] Margineantu, D., T. Dietterich. 1997. Pruning adaptive boosting. *Proceedings of the Fourteenth International Conference Machine Learning*.

[17] Meinshausen, N. 2010. Node harvest. *The Annals of Applied Statistics* 2049–2072.

[18] Nijssen, S., E. Fromont. 2007. Mining optimal decision trees from itemset lattices. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[19] Ribeiro, M.T., S. Singh, C. Guestrin. 2016. "Why Should I Trust You?" Explaining the Predictions of Any Classifier. *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD '16*. 1135–1144.

[20] Rokach, L. 2016. Decision forest: Twenty years of research. *Information Fusion* **27** 111–125.

Bibliography III

[21] Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* **1**(5) 206–215.

[22] Serra, T., A. Kumar, S. Ramalingam. 2020. Lossless compression of deep neural networks. *CPAIOR 2020: Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. 417–430.

[23] Tamon, C., J. Xiang. 2000. On the boosting pruning problem. *Proceedings of the 11th European Conference on Machine Learning*.

[24] Tan, H. F., G. Hooker, M. T. Wells. 2016. Tree space prototypes: Another look at making tree ensembles interpretable. *arXiv preprint arXiv:1611.07115* .

[25] Verwer, S., Y. Zhang. 2019. Learning optimal classification trees using a binary linear program formulation. *Proceedings of the AAAI Conference on Artificial Intelligence*.

[26] Zhang, Y., S. Burer, W. N. Street. 2006. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research* **7**(Jul) 1315–1338.