



Universitat de Lleida



MONASH University



ICREA



reason: Efficient Explanations for Rule Ensembles

Hao Hu¹, Alexey Ignatiev², Joao Marques-Silva³

¹University of Lleida, Lleida, Spain

²Monash University, Melbourne, Australia

³ICREA, University of Lleida, Lleida, Spain

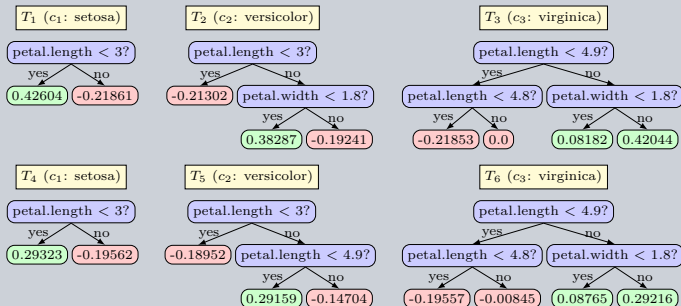
- The natural need of eXplainable AI (XAI).
 - ▶ Improve the **explainability** and **trustworthy** of predictions made by complex AI system.
- Growing interest in **symbolic XAI** (Formal XAI) for the model-aware post-hoc explanations, especially, for **Tree Ensembles** (TEs) model.
 - ▶ v.s. model-agnostic approaches → symbolic XAI generates logically rigorous explanations representing the model's semantics.
 - ▶ *Why TEs?* TEs are based on tree structures with solid predictive performance.
- Current scalability limitation of SOTAs for computing formal explanations of TEs.
 - ▶ TEs have complicated structures.
- **Rule Ensembles** (REs): an alternative of TEs.
 - ▶ v.s. TEs → widely competitive predictive performance + more compact model structures.
 - ▶ No existing method to compute formal explanations for REs.
 - ▶ Reduction in model complexity leads the improvement in scalability of formal explanation → Can we handle image dataset? (MNIST, yes!)

- **Abductive Explanations (AXps)** explains *Why* a prediction for an instance is made.
 - ▶ subset-minimal feature-value pairs that are sufficient for the given prediction.
- For a classifier $\tau : \mathbb{F} \rightarrow \mathcal{K}$ and a given instance $(\mathbf{v}, c_p)$ that $c_p = \tau(\mathbf{v})$, an AXp is a minimal subset $\mathcal{X} \subseteq \mathbb{F}$ such that:

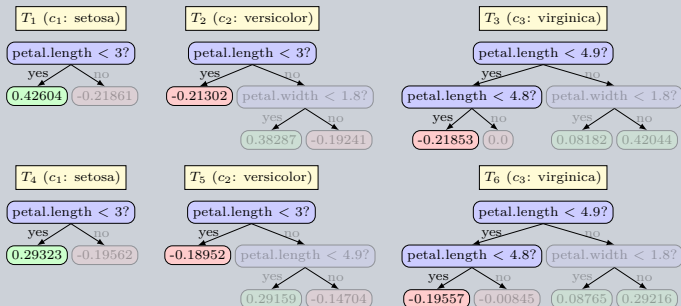
$$\forall(\mathbf{x} \in \mathcal{F}). [\bigwedge_{j \in \mathcal{X}} (x_j = v_j)] \rightarrow (\tau(\mathbf{x}) = c_p)$$

- For a **Tree Ensemble (TE)** dealing classification with **K** classes, of **N** trees per class.
 - ▶ The t -th tree in TE is $\mathcal{T}_t : \mathbb{F} \rightarrow \mathbb{R}$, mapping $\mathbf{x} \in \mathbb{F}$ to the weight associating the leaf lead to.
 - ▶ The sum of weight for class c_i is $w_i^t(\mathbf{x}) = \sum_{j \in \{0, \dots, N-1\}} \mathcal{T}_{Kj+i}(\mathbf{x}), i \in [K]$.
 - ▶ The prediction of TE is $\tau(\mathbf{x}) = \arg \max_{i \in [K]} w_i^t(\mathbf{x})$.
- For a **Rule Ensemble (RE)** with the same condition.
 - ▶ A default rule $\mathcal{R}_0$ contains K *default weights* for per class, w_i^{r0} is the default weight for c_i .
 - ▶ The t -th ($t \neq 0$) rule in RE is $\mathcal{R}_t : \mathbb{F} \rightarrow \mathbb{R}$, mapping $\mathbf{x} \in \mathbb{F}$ to the weight.
 - ▶ The sum of class c_i is $w_i^r(\mathbf{x}) = w_i^{r0} + \sum_{j \in \{0, \dots, N-1\}} \mathcal{R}_{Kj+i}(\mathbf{x}), i \in [K]$.
 - ▶ The prediction of RE is $\tau(\mathbf{x}) = \arg \max_{i \in [K]} w_i^r(\mathbf{x})$.

- An illustrative TE (based on XGBoost) for Iris dataset ($N = 2, K = 3$).

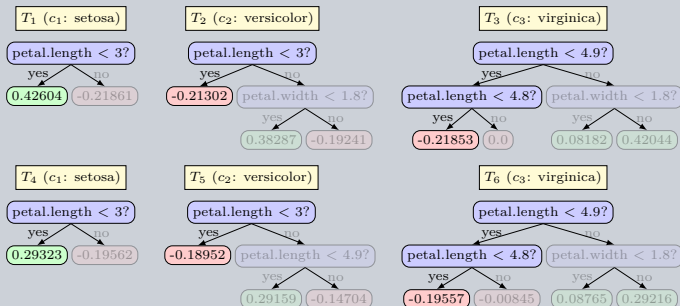


- An illustrative TE (based on XGBoost) for Iris dataset ($N = 2, K = 3$).



- For a given instance
 $((sepal.length = 5.1) \wedge (sepal.width = 3.5) \wedge (petal.length = 1.4) \wedge (petal.width = 0.2))$
 the prediction of TE is c_1 as shown in the trees lighted.

- An illustrative TE (based on XGBoost) for Iris dataset ($N = 2, K = 3$).



- For a given instance $((sepal.length = 5.1) \wedge (sepal.width = 3.5) \wedge (petal.length = 1.4) \wedge (petal.width = 0.2))$ the prediction of TE is c_1 as shown in the trees lighted.
- The AXp of the illustrative TE for the given instance is clearly the $\{(petal.length = 1.4)\}$.

- Respectively, an illustrative RE (based on Boomer) for Iris dataset ($N = 2, K = 3$).

R_0 : **DEFAULT** c_1 : setosa = -0.31356, c_2 : versicolor = -0.29661, c_3 : virginica = -0.38136

R_1 : **IF** petal.length $\leq$ 2.45 **THEN** c_1 : setosa = 0.38446

R_2 : **IF** petal.length $\leq$ 2.45 **THEN** c_2 : versicolor = -0.20587

R_3 : **IF** petal.width $>$ 1.75 **THEN** c_3 : virginica = 0.38542

R_4 : **IF** petal.length $>$ 2.45 **THEN** c_1 : setosa = -0.20329

R_5 : **IF** petal.length $>$ 5.05 **AND** sepal.width $>$ 2.75 **THEN** c_2 : versicolor = -0.20348

R_6 : **IF** petal.width $>$ 1.75 **THEN** c_3 : virginica = 0.27310

- Respectively, an illustrative RE (based on Boomer) for Iris dataset ($N = 2, K = 3$).

R_0 : DEFAULT c_1 : setosa = -0.31356, c_2 : versicolor = -0.29661, c_3 : virginica = -0.38136
 R_1 : IF petal.length $\leq$ 2.45 THEN c_1 : setosa = 0.38446
 R_2 : IF petal.length $\leq$ 2.45 THEN c_2 : versicolor = -0.20587
 R_3 : IF petal.width $>$ 1.75 THEN c_3 : virginica = 0.38542
 R_4 : IF petal.length $>$ 2.45 THEN c_1 : setosa = -0.20329
 R_5 : IF petal.length $>$ 5.05 AND sepal.width $>$ 2.75 THEN c_2 : versicolor = -0.20348
 R_6 : IF petal.width $>$ 1.75 THEN c_3 : virginica = 0.27310

- For a given instance
 $((sepal.length = 5.1) \wedge (sepal.width = 3.5) \wedge (petal.length = 1.4) \wedge (petal.width = 0.2))$
 the prediction of RE is c_1 as shown in the rules underlined.

- Respectively, an illustrative RE (based on Boomer) for Iris dataset ($N = 2, K = 3$).

R_0 :	DEFAULT	c_1 : setosa = -0.31356, c_2 : versicolor = -0.29661, c_3 : virginica = -0.38136
R_1 :	IF petal.length $\leq$ 2.45 THEN	<u>c_1 : setosa = 0.38446</u>
R_2 :	IF petal.length $\leq$ 2.45 THEN	<u>c_2 : versicolor = -0.20587</u>
R_3 :	IF petal.width $>$ 1.75 THEN	<u>c_3 : virginica = 0.38542</u>
R_4 :	IF petal.length $>$ 2.45 THEN	<u>c_1 : setosa = -0.20329</u>
R_5 :	IF petal.length $>$ 5.05 AND sepal.width $>$ 2.75 THEN	c_2 : versicolor = -0.20348
R_6 :	IF petal.width $>$ 1.75 THEN	c_3 : virginica = 0.27310

- For a given instance
 $((sepal.length = 5.1) \wedge (sepal.width = 3.5) \wedge (petal.length = 1.4) \wedge (petal.width = 0.2))$
 the prediction of RE is c_1 as shown in the rules underlined.
- The AXp of the illustrative RE for the given instance is also the $\{(petal.length = 1.4)\}$.

Function ExtractAXp($\Re e, (\mathbf{v}, c_p)$):

Input: $\Re e$: RE computing the $\tau(\mathbf{v}), (\mathbf{v}, c_p)$:

Input instance

Output: $\mathcal{X}$: abductive explanation

// An overestimated explanation $\mathcal{X}$

1 $\mathcal{X} \leftarrow \mathcal{F}$

// Encode RE via an automated
reasoning approach

2 $\langle \mathcal{H} \rangle \leftarrow \text{Encode}(\Re e)$

3 **for** $j \in \mathcal{X}$ **do**

 // Entailment check for the need
 of j .

4 **if** EntCheck($\mathcal{H}, (\mathbf{v}, c_p), \mathcal{X} \setminus \{j\}$) **then**

5 $\mathcal{X} \leftarrow \mathcal{X} \setminus \{j\}$

6 **end**

7 **end**

// $\mathcal{X}$ is an AXp

8 **return** $\mathcal{X}$

Function $\text{ExtractAXp}(\mathcal{R}e, (\mathbf{v}, c_p))$:

Input: $\mathcal{R}e$: RE computing the $\tau(\mathbf{v})$, $(\mathbf{v}, c_p)$:

Input instance

Output: $\mathcal{X}$: abductive explanation

// An overestimated explanation $\mathcal{X}$

1 $\mathcal{X} \leftarrow \mathcal{F}$

// Encode RE via an automated reasoning approach

2 $\langle \mathcal{H} \rangle \leftarrow \text{Encode}(\mathcal{R}e)$

3 **for** $j \in \mathcal{X}$ **do**

 // Entailment check for the need of j .

4 **if** $\text{EntCheck}(\mathcal{H}, (\mathbf{v}, c_p), \mathcal{X} \setminus \{j\})$ **then**

5 $\mathcal{X} \leftarrow \mathcal{X} \setminus \{j\}$

6 **end**

7 **end**

// $\mathcal{X}$ is an AXp

8 **return** $\mathcal{X}$

- Two parts of constraints to encode a RE:
 - ▶ Constraints for **RE operations**.
 - ▶ Constraints **enforcing the winning class**.

Function ExtractAXp($\mathcal{R}e, (\mathbf{v}, c_p)$):

Input: $\mathcal{R}e$: RE computing the $\tau(\mathbf{v})$, $(\mathbf{v}, c_p)$:
Input instance

Output: $\mathcal{X}$: abductive explanation

// An overestimated explanation $\mathcal{X}$

$\mathcal{X} \leftarrow \mathcal{F}$

// Encode RE via an automated
reasoning approach

$\langle \mathcal{H}, \mathcal{S} \rangle \leftarrow \text{EncodeMX}(\mathcal{R}e)$

for $j \in \mathcal{X}$ **do**

 // Entailment check for the need
 of j .

if $\text{EntCheckMX}(\mathcal{H}, \mathcal{S}, (\mathbf{v}, c_p), \mathcal{X} \setminus \{j\})$
 then

$\mathcal{X} \leftarrow \mathcal{X} \setminus \{j\}$

end

end

// $\mathcal{X}$ is an AXp

return $\mathcal{X}$

- Two parts of constraints to encode a RE:

- ▶ Constraints for **RE operations**.
- ▶ Constraints **enforcing the winning class**.

- **MaxSAT encoding:**

- ▶ Idea referred to MaxSAT encoding for TEs.

- ▶ For any class

$$\max \quad S_{l,p} = \sum_l - \sum_p$$

$$\text{s.t.} \quad \mathcal{H} \wedge \bigwedge_{j \in \mathcal{X}} I[x_j = v_j]$$

- ▶ $\mathcal{H}$: encoding RE operations.
- ▶ $\max(S_{l,p}) < 0$: enforcing the winning class as c_p not c_l .
- ▶ MaxSAT oracles are optimization problems, not decision problems!

Function ExtractAXp($\Re e, (\mathbf{v}, c_p)$):

Input: $\Re e$: RE computing the $\tau(\mathbf{v})$, $(\mathbf{v}, c_p)$:
Input instance

Output: $\mathcal{X}$: abductive explanation

// An overestimated explanation $\mathcal{X}$

$\mathcal{X} \leftarrow \mathcal{F}$

// Encode RE via an automated
reasoning approach

$\langle \mathcal{H}, \mathcal{H}_{l,p}^{pred} \rangle \leftarrow \text{EncodeSAT}(\Re e)$

for $j \in \mathcal{X}$ **do**

 // Entailment check for the need
 of j .

if $\text{EntCheckSAT}(\mathcal{H}, \mathcal{H}_{l,p}^{pred}, (\mathbf{v}, c_p), \mathcal{X} \setminus$
 $\{j\})$ **then**

$\mathcal{X} \leftarrow \mathcal{X} \setminus \{j\}$

end

end

// $\mathcal{X}$ is an AXp

return $\mathcal{X}$

- Two parts of constraints to encode a RE:

- ▶ Constraints for **RE operations**.
- ▶ Constraints **enforcing the winning class**.

• MaxSAT encoding:

• SAT encoding:

- ▶ The **negation** of $\max(S_{l,p}) < 0$ is to satisfy:

$$\mathcal{H}_{l,p}^{pred} \wedge \mathcal{H} \wedge \bigwedge_{j \in \mathcal{X}} I[x_j = v_j]$$

- ▶ $\mathcal{H}_{l,p}^{pred}$: the Pseudo-Boolean constraint $(\sum_l - \sum_p) \geq 0$.
- ▶ If any oracles satisfied, the *EntCheck* failed.
- ▶ Transform optimization problems to SAT decision problems.

- Encodings adapted from xreason, the MaxSAT encoding to explain TEs.
- $\mathcal{F}$: set of features; $\forall_{j \in \mathcal{F}} D_j$: domain of feature x_j ; $\mathfrak{Re}$: rule ensemble.
- The constraints encoding RE operations $\mathcal{H} = \bigwedge_{j \in \mathcal{F}} \mathcal{H}_{D_j} \wedge \bigwedge_{\mathcal{R}_i \in \mathfrak{Re}} \mathcal{H}_{\mathcal{R}_i}$.

- $\mathcal{H}_{D_j}$ encodes **feature domain** D_j .
 - ▶ feature threshold values $s_{j,k}$ from $\mathfrak{Re}$.
 - ▶ Bool variable $o_{j,k} = 1$ iff $x_j \leq s_{j,k}$.
 - ▶ Bool variable $l_{j,k} = 1$ iff $x_j \in (s_{j,k-1}, s_{j,k}]$.
 - ▶ **order encoding** of D_j with $l_{j,k}$ and $o_{j,k}$.

- $\mathcal{H}_{\mathcal{R}_i}$ encodes **conditions in rule** $\mathcal{R}_i$.
 - ▶ Bool variable $h_i = 1$ iff head of rule $\mathcal{R}_i$ is reached.
 - ▶ The head of default rule $\mathcal{R}_0$ is always reached: $h_0 = 1$.
 - ▶ p_i denotes the conditions in rule $\mathcal{R}_i$.
 - ▶ Encode the conditions of each rule:

$$\left(\bigwedge_{x_j \leq s_{j,k} \in p_r} o_{j,k} \wedge \bigwedge_{x_j > s_{j,k} \in p_r} \neg o_{j,k} \right) \leftrightarrow h_r$$

Function EntCheckMX($\mathcal{H}, \mathcal{S}, (\mathbf{v}, c_p), \mathcal{X}$):

Input: $\mathcal{H}$: Hard clauses, $\mathcal{S}$: Soft clauses,
 $(\mathbf{v}, c_p)$: Input instance,
 $\mathcal{X}$: Candidate explanation

Output: *true* or *false*

```

1  for  $S_{l,p} \in \mathcal{S}$  do
    //  $S_{l,p}$  are soft clauses
    //   relevant to  $c_l$  and  $c_p$ 
2     $\mu = \text{MaxSAT}(\mathcal{H} \wedge \bigwedge_{j \in \mathcal{X}} I[x_j =$ 
       $v_j], S_{l,p})$ 
3    if ObjValue( $\mu$ )  $\geq 0$  then
4      return false
5    end
6  end
7  return true
  
```

- $(K - 1)$ MaxSAT oracles:
 - ▶ Each MaxSAT oracle determine the objective value of $S_{l,p} = (\sum_l - \sum_p)$.
 - ▶ Soft clauses $S_{l,p}$ are **weighted soft clause** (h_r with real value weights).
 - ▶ $\mathcal{X}$ is no longer a valid AXp if $S_{l,p} \geq 0$.
 - ▶ MaxSAT oracle solved by core-guided MaxSAT solvers with assumptions.

Function EntCheckSAT($\mathcal{H}, \mathcal{H}^{pred}, (\mathbf{v}, c_p), \mathcal{X}$):

Input: $\mathcal{H}$: Hard clauses, $\mathcal{H}^{pred}$: PB constraints, $(\mathbf{v}, c_p)$: Input instance,

$\mathcal{X}$: Candidate explanation

Output: *true* or *false*

```

1  for  $\mathcal{H}_{l,p}^{pred} \in \mathcal{H}^{pred}$  do
    //  $\mathcal{H}_{l,p}^{pred}$  is the PB constraint
    for  $(\sum_l - \sum_p) \geq 0$ 
2      if SAT( $\mathcal{H}_{l,p}^{pred} \wedge \mathcal{H} \wedge \bigwedge_{j \in \mathcal{X}} I[x_j = v_j]$ )
3          then
4              return false
5      end
6  end
  return true

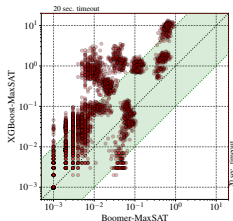
```

- $(K - 1)$ SAT oracles:
 - ▶ Each SAT oracle decide the existence that $(\sum_l - \sum_p) \geq 0$.
 - ▶ The usage of PB Constraints.
- However, PB constraints only accept **integer coefficient** $\rightarrow$ mapping real value weights to integer weights.
 - ▶ Preserving small number (i.e w) of decimal places for all real value weights.
 - ▶ Integer weights can be got via multiplying 10^w for all real value weights.
 - ▶ Greatest common divisor can be used to further reduce the integer weights.

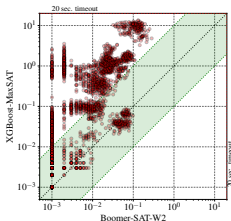
- **Machine configuration:** MacBook Pro running M4 CPU with 24GB memory.
- **Tools configurations:** Prototype¹ is realized in Python.
 - ▶ Formula preparation: PySAT toolkit.
 - ▶ Solvers: RC2 solver for MaxSAT, Glucose 3 solver for SAT.
 - ▶ PB Constraints encoding: best encoding selected by PBLib.
- **Tabular benchmarks:** UCI ML Repository + Penn ML Repository (21 datasets)
 - ▶ 7 – 60 features, 26.95 on average.
 - ▶ 59 – 58000 data instances, 4684.47 on average.
 - ▶ For each dataset, (80%, 20%) as the training/testing split ratio.
 - ▶ 200 randomly selected instances to explain for per dataset.
- **Image benchmark:** MNIST dataset
 - ▶ $28 \times 28 = 784$ features, 60000 data instances.
 - ▶ (80%, 20%) as the training/testing split ratio.
 - ▶ 100 randomly selected instances to explain.

¹<https://github.com/nwpuhh/yreason>

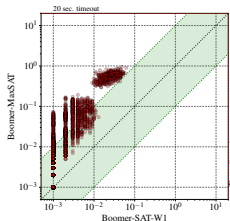
- **Ensemble models to explain:** $\text{test_acc} \geq 0.95$ with minimal number of unique rules.
 - ▶ Number of unique rules (or paths in trees) can represent the size of REs and TEs.
 - ▶ TEs learned by XGBoost (max_depth 4 – 6, number of trees per class 10 – 100) → average number of unique paths is **400.9**, average **train_acc** = **0.989**, average **test_acc** = **0.933**.
 - ▶ REs learned by Boomer (max_cond 4 – 6, number of rules per class 10 – 100) → average number of unique paths is **129.7**, average **train_acc** = **0.979**, average **test_acc** = **0.931**.
 - ▶ Two precision-reduced REs, no change in RE structure:
 - RE-W2 (2 decimal places preserved) → average **train_acc** = **0.978**, average **test_acc** = **0.930**.
 - RE-W1 (1 decimal places preserved) → average **train_acc** = **0.977**, average **test_acc** = **0.923**.
 - ▶ REs are more compact than TEs, also with competitive predictive performance, even with precision reduction.
- **Methods to evaluate:** in the efficiency in computing AXps.
 - ▶ **XGBoost-MaxSAT (SOTA):** The MaxSAT approach computing an AXp for TEs.
 - ▶ **Boomer-MaxSAT:** The proposed MaxSAT approach computing an AXp for REs.
 - ▶ **Boomer-SAT-W*:** The proposed SAT approach computing an AXp for precision-reduced REs, * indicates the number of decimal places preserved.



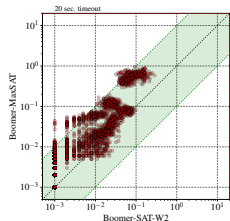
(a) Boomer-MaxSAT vs. XGBoost-MaxSAT



(b) Boomer-SAT-W2 vs. XGBoost-MaxSAT



(c) Boomer-SAT-W1 vs. Boomer-MaxSAT



(d) Boomer-SAT-W2 vs. Boomer-MaxSAT

- **A brief summary in execution time for per instance (200 in total)**

- ▶ XGBoost-MaxSAT: 0.001 – 13.58 seconds, average 0.72 seconds.
- ▶ Boomer-MaxSAT: 0.001 – 1.18 seconds, average 0.09 seconds.
- ▶ Boomer-SAT-W2: 0.001 – 0.18 seconds, average 0.02 seconds.
- ▶ Boomer-SAT-W1: 0.001 – 0.04 seconds, average 0.01 seconds.

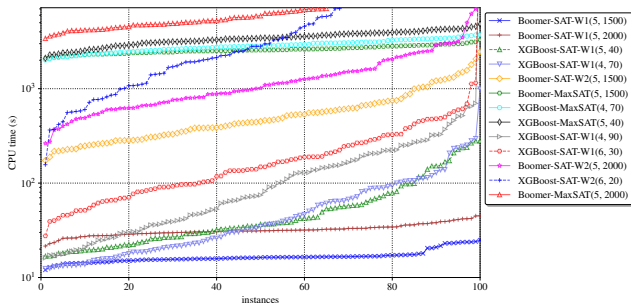
- The proposed methods computing AXps for REs are about **up to 2 order of magnitude** to the SOTA in runtime per instance.

- Ensemble models to explain: **test_acc** ≥ 0.95 .
 - ▶ Number of unique paths (model size) $\leq 20K$.
 - ▶ All REs and TEs are in format of *Method-(md, n)* indicating model, maximum depth (or condition), and the number of trees (or rules).
 - ▶ *Model-W** report the information of precision-reduced variants.

Model-(md,n)	Model Size	Orig		Model-W1		Model-W2	
		train	test	train	test	train	test
XGBoost-(4, 70)	10145	0.993	0.966	0	0.001	0	0
XGBoost-(6, 20)	10444	0.987	0.957	0	0	0	0
XGBoost-(5, 40)	10984	0.993	0.965	0	0	0	0.001
XGBoost-(4, 80)	11501	0.996	0.968	0	-0.001	0	0
XGBoost-(4, 90)	12852	0.998	0.969	-0.001	-0.001	0	0
XGBoost-(4, 100)	14199	0.999	0.970	0	-0.001	0	0
XGBoost-(6, 30)	14587	0.996	0.965	0	0	0	0
Boomer-(6, 1500)	14823	0.990	0.956	-0.018	-0.023	0	0
Boomer-(4, 1500)	14843	0.979	0.950	-0.046	-0.047	0	0
Boomer-(5, 1500)	14860	0.985	0.954	-0.029	-0.030	0	0
Boomer-(6, 2000)	19784	0.994	0.958	-0.018	-0.023	0	0.001
Boomer-(4, 2000)	19793	0.984	0.953	-0.047	-0.049	0.001	0
Boomer-(5, 2000)	19827	0.989	0.956	-0.028	-0.031	0	0.001

- The *Model-W2* variants have no lose in prediction, limited lose for *Boomer-W1* models.

- Time limit to explain an instance: 7200 seconds; 100 randomly selected instances.
- Summary in execution time for per instance (≥ 60 instances solved in time limit):
 - ▶ Boomer-SAT-W1: 10.36 - 94.72 seconds, average 24.30 seconds.
 - ▶ XGBoost-SAT-W1: 12.44 - 4768.34 seconds, average 144.03 seconds.
 - ▶ Boomer-SAT-W2: 146.23 - 7009.46 seconds, average 1063.93 seconds.
 - ▶ XGBoost-SAT-W2: failed all except XGBoost-SAT-W2(6, 20), average 2278.38 seconds.
 - ▶ Boomer-MaxSAT: 1593.22 - 7147.84 seconds, average 3286.08 seconds.
 - ▶ XGBoost-MaxSAT: 2017.12 - 7198.28 seconds, average 3772.73 seconds.



- The SAT encoding outperforms a lot than MaxSAT encoding.

- An adapting efficient MaxSAT-based approach to formally explain Rule Ensembles.
 - A dedicated encoding combining SAT solving with pseudo-Boolean (PB) constraints to improve the efficiency.
 - Empirical experiments in both tabular and image datasets demonstrating the efficiency of AXps extraction via SAT-based approach for REs.
-
- Critical bottleneck: the growth of PB constraint size.
 - ▶ Direction 1: developing dedicated PB constraints encoding to tailor the explanation for REs.
 - ▶ Direction 2: proposing new mechanisms for SAT solvers to handle large PB constraints.

Thank you!