POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

ICML 2024 - Vienna, Austria (July 24, 2024)

Trained Random Forests Completely Reveal your Dataset

Julien Ferry[1], Ricardo Fukasawa[2], Timothée Pascal[3], Thibaut Vidal[1]

[1]Polytechnique Montréal, Canada
[2]University of Waterloo, Canada
[3]Ecole Nationale des Ponts et Chaussées, Paris, France

julien.ferry@polymtl.ca

Table of Contents

1. Background
2. Illustrative Example: Reconstruction from Tree-Based Models
3. DRAFT: Dataset Reconstruction from Tree Ensembles
4. Experimental Study
5. Discussion

Table of Contents

1. **Background**
   - Notations & Supervised Learning
   - Random Forests
   - Dataset Reconstruction Attacks
2. Illustrative Example: Reconstruction from Tree-Based Models
3. DRAFT: Dataset Reconstruction from Tree Ensembles
4. Experimental Study
5. Discussion

Notations & Supervised Machine Learning

Notation

- Consider a labeled dataset $\mathcal{D} = \{\mathrm{x}_k; c_k\}_{k=1}^{N}$
  - Each example $k \in \{1..N\}$ is characterized by a vector $\mathrm{x}_k \in \{0,1\}^M$ of M binary attributes and a class $c_k \in \mathcal{C}$
  - *vects* is the list of the different groups (if any) of binary attributes one-hot encoding the same original feature

Table 1: Example binary dataset $\mathcal{D}$ with $N = 4$ and $M = 4$.

f_1	f_2	f_3	f_4	c
0	0	0	1	0
1	0	0	0	0
0	1	0	0	1
1	0	1	1	1

Decision Trees

- A tree t is made of:
  - a set of internal nodes $\mathcal{V}_t^I$
  - a set of leaves $\mathcal{V}_t^L$
- For every node $v \in \mathcal{V}_t^I \bigcup \mathcal{V}_t^L$, let nb_{tvc} denote the number of training examples of class c that went through v
- For every leaf $v \in \mathcal{V}_t^L$, we let $\Phi_v \subseteq \{1..M\}$ (*resp.*, $\overline{\Phi_v} \subseteq \{1..M\}$) denote the set of indices of the binary attributes that must be 1 (*resp.*, 0) for an example to fall into v

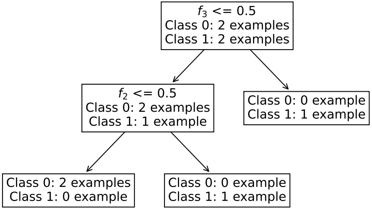


Figure 1: Example decision tree t_1 trained on $\mathcal{D}$ (Table 1).

Training Random Forests

- A Random Forest $\mathcal{T}$ is a set of Decision Trees
- To encourage diversity between the different trees, several randomization mechanisms are used during training:
  - **Bagging (Bootstrap AGGgregatING)** consists in generating different training sets (one for each $t \in \mathcal{T}$) using sampling with replacement (usually from N to N examples)

Reconstruction Attacks

- **Reconstruction Attacks** are a type of inference attack first proposed against database access mechanisms [Dinur and Nissim, 2003]
- The objective is to retrieve (entirely or partly) the dataset used as input of an algorithm, leveraging only its output(s)

Considered Setup

- An adversary has white-box access to a Random Forest $\mathcal{T}$ trained on $\mathcal{D}$
- The adversary wants to reconstruct a dataset $\mathcal{D}'$ as close as possible to $\mathcal{D}$

Table of Contents

1. Background
2. **Illustrative Example: Reconstruction from Tree-Based Models**
3. DRAFT: Dataset Reconstruction from Tree Ensembles
4. Experimental Study
5. Discussion

Example Reconstruction from a Decision Tree

Table 2: Example binary dataset $\mathcal{D}$ with $N = 4$ and $M = 4$.

f_1	f_2	f_3	f_4	c
0	0	0	1	0
1	0	0	0	0
0	1	0	0	1
1	0	1	1	1

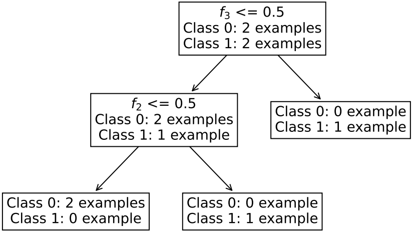


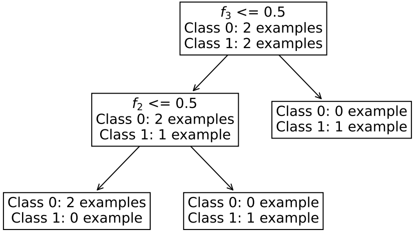
Figure 2: Example decision tree t_1 trained on $\mathcal{D}$ (Table 2).

Example Reconstruction from a Decision Tree

Table 2: Example binary dataset $\mathcal{D}$ with $N = 4$ and $M = 4$.

f_1	f_2	f_3	f_4	c
0	**0**	**0**	1	**0**
1	**0**	**0**	0	**0**
0	**1**	**0**	0	**1**
1	0	**1**	1	**1**

Bold values can be reconstructed from Figure 2's decision tree t_1.



f_3 <= 0.5
Class 0: 2 examples
Class 1: 2 examples

f_2 <= 0.5
Class 0: 2 examples
Class 1: 1 example

Class 0: 0 example
Class 1: 1 example

Class 0: 2 examples
Class 1: 0 example

Class 0: 0 example
Class 1: 1 example

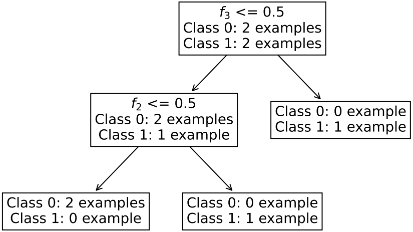
Figure 2: Example decision tree t_1 trained on $\mathcal{D}$ (Table 2).

Example Reconstruction from a Decision Tree

Table 2: Example binary dataset $\mathcal{D}$ with $N = 4$ and $M = 4$.

f_1	f_2	f_3	f_4	c
0	**0**	**0**	1	**0**
1	**0**	**0**	0	**0**
0	**1**	**0**	0	**1**
1	0	**1**	1	**1**

Bold values can be reconstructed from Figure 2's decision tree t_1.



f_3 <= 0.5
Class 0: 2 examples
Class 1: 2 examples

f_2 <= 0.5
Class 0: 2 examples
Class 1: 1 example

Class 0: 0 example
Class 1: 1 example

Class 0: 2 examples
Class 1: 0 example

Class 0: 0 example
Class 1: 1 example

Figure 2: Example decision tree t_1 trained on $\mathcal{D}$ (Table 2).

⇒ This reconstruction process was introduced by Gambs et al. [2012] (for a single decision tree)

Reconstruction from a Random Forest: Some Challenges

Challenges

- How can we **match** the information provided by different trees ?
- What if **bagging** is used (and so some examples may be used multiple times to train a tree, and others not at all) ?

Table of Contents

1. Background

2. Illustrative Example: Reconstruction from Tree-Based Models

3. DRAFT: Dataset Reconstruction from Tree Ensembles
   - Principle
   - Constraint Programming Formulation

4. Experimental Study

5. Discussion

Principle

Reconstructing a Random Forest's Training Set

- It is a NP-Complete problem
- We propose Mixed-Integer Linear Programming and **Constraint Programming** formulations encoding the structure of the trees within the forest
  - Each tree defines a set of constraints
  - We leverage general purpose solvers to find feasible reconstructions given the provided constraints
- When bagging is used, we use variables modelling the bootstrap sampling process and maximize the likelihood of their assignments

Constraint Programming Formulation: Variables

We define variables: (i) modelling the reconstructed examples' attributes and class label, (ii) assigning the examples to the trees' leaves, and (iii) encoding the bagging process.

- $z_{kc} \in \{0;1\}$ is 1 if training example k is part of class c, 0 otherwise
- $x_{ki} \in \{0;1\}$ is the value of feature i for example k in the reconstruction
- $y_{tvkc} \in \mathbb{Z}_+$ is the number of times training example k is classified by leaf v within tree t as class c
- $q_{tkb} \in \{0;1\}$ is 1 if training example k is used $b \in \mathcal{B}$ times in tree t and 0 otherwise

NB: $\mathcal{B}$ is the set of possible values for the number of occurences of an example within a tree's training set (theoretically, $\mathcal{B} = \{1..N\}$).

Constraint Programming Formulation II

Constraint Programming Formulation: Constraints

- $\forall k \in \{1..N\}, \forall w \in vects$: $\sum_{i \in w} x_{ki} = 1$ ← One-Hot Encoding Consistency
- $\forall k \in \{1..N\}$: $\sum_{c \in \mathcal{C}} z_{kc} = 1$ ← Each example is assigned to exactly one class
- $\forall k \in \{1..N\}, \forall c \in \mathcal{C}$: **if** $z_{kc} = 0$ **then** $\sum_{t \in \mathcal{T}, v \in \mathcal{V}_t^L} y_{tvkc} = 0$ ← Example appears only in its class' counts
- $\forall t \in \mathcal{T}, \forall v \in \mathcal{V}_t^L, \forall c \in \mathcal{C}$: $nb_{tvc} = \sum_{k \in \{1..N\}} y_{tvkc}$ ← The counts match the number of assigned examples
- $\forall t \in \mathcal{T}, \forall k \in \{1..N\}, \forall v \in \mathcal{V}_t^L$:

  if $\sum_{c \in \mathcal{C}} y_{tvkc} \geq 1$ **then** $\left(\bigwedge_{i \in \Phi_v} x_{ki} = 1 \right) \wedge \left(\bigwedge_{i \in \overline{\Phi_v}} x_{ki} = 0 \right)$ ← The attributes' values are consistent with the splits
- $\forall t \in \mathcal{T}, \forall b \in \mathcal{B}, \forall k \in \{1..N\}$: $\sum_{v \in \mathcal{V}_t^L, c \in \mathcal{C}} y_{tvkc} = b \iff q_{tkb} = 1$ ← One hot encodes the #occurrences

Constraint Programming Formulation III

Constraint Programming Formulation: Objective

- The defined formulation could have many possible solutions, so we orient the search towards the most likely ones
- Bagging performs sampling with replacement from N examples to N examples
$\Rightarrow$ The probability that an example appears exactly b times in a tree's training set is then:

$$p_b = \left(\frac{1}{N}\right)^b \cdot \left(\frac{N-1}{N}\right)^{N-b} \cdot \binom{N}{b}$$

- The overall probability of a given assignment of the bagging variables $\left\{q_{t\in\mathcal{T}\,k\in\{1..N\}\,b\in\mathcal{B}}\right\}$ is then:

$$\prod_{t\in\mathcal{T}} \prod_{k\in\{1..N\}} p_{b|q_{tkb}=1}$$

- Maximizing this probability is equivalent to maximizing its logarithm, so we maximize:

$$\sum_{t\in\mathcal{T}} \sum_{k\in\{1..N\}} \sum_{b\in\mathcal{B}} \log\left(p_b\right) q_{tkb} \leftarrow \textbf{Our objective function}$$

Table of Contents

1. Background
2. Illustrative Example: Reconstruction from Tree-Based Models
3. DRAFT: Dataset Reconstruction from Tree Ensembles
4. **Experimental Study**
   - Experimental Setup
   - Results
   - Other Takeaways
5. Discussion

1) Random Forests Training

- We learn Random Forests using the `scikit-learn` Python library [Pedregosa et al., 2011]:
  - With or without Bagging
  - With $|\mathcal{T}| \in \{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$
  - With maximum tree depth $d_{max} \in \{\text{None}, 2, 3, 4, 5, 10\}$
  - On three datasets (from which we randomly subsample a training set with $N = 100$):

Dataset	Binary Prediction Task	#Examples	#Binary Features
COMPAS [Angwin et al., 2016]	Recidivism prediction	$7,206$	15
UCI Adult Income [Dua and Graff, 2017]	Income $>$ \$50K/year	$48,842$	20
Default of Credit Card Client [Yeh and hui Lien, 2009]	Default in payment	$29,986$	22

Experimental Setup II

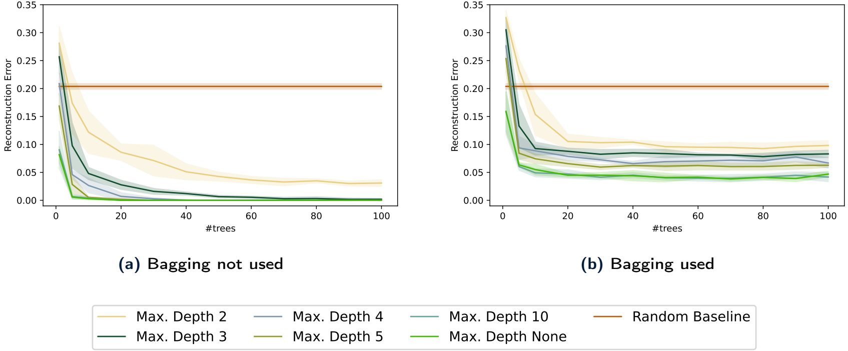
2) Reconstruction Attack

- We use the `OR-Tools` CP-SAT solver [Perron and Didier, 2023] to solve our CP formulations, with each run being limited to 5 hours of CPU time
- Baseline: random reconstruction, aware of one-hot encoding
  - Same knowledge as ours (number of examples N, of attributes M, and their one-hot encoding *vects*) except for the Random Forest itself

3) Attack Success Evaluation

- Reconstruction error computation:
  - We first align the original and reconstructed datasets using a minimum cost matching
  - We then compute the average Manhattan distance between the original examples and their matched reconstructed counterparts

Results I

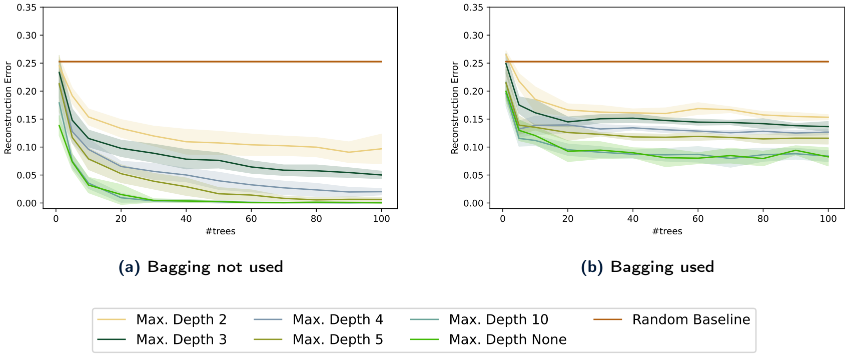


(a) Bagging not used

(b) Bagging used

Figure 3: Average reconstruction error as a function of the number of trees $|\mathcal{T}|$ within the target forest $\mathcal{T}$, for different maximum depth values d_{max} and for the random baseline, COMPAS dataset.

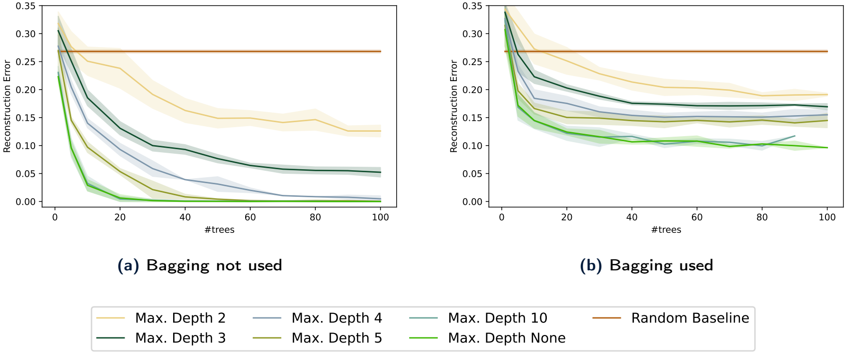
Results II



(a) Bagging not used

(b) Bagging used

Figure 4: Average reconstruction error as a function of the number of trees $|\mathcal{T}|$ within the target forest $\mathcal{T}$, for different maximum depth values d_{max} and for the random baseline, UCI Adult Income dataset.



(a) Bagging not used

(b) Bagging used

Figure 5: Average reconstruction error as a function of the number of trees $|\mathcal{T}|$ within the target forest $\mathcal{T}$, for different maximum depth values d_{max} and for the random baseline, Default of Credit Card Client dataset.

Other Experimental Takeaways

- The use of Bagging makes the reconstruction task more difficult . . .
  - . . . but if the adversary correctly guesses the number of occurrences of each example within each tree's training set, the reconstruction accuracy is as good as in the non-bagging case (or even better)
- Partial knowledge of the training set can help reconstruct the unknown part
  - Knowledge of some attributes (columns) helps reconstruct the others
  - Knowledge of some examples (rows) does not help reconstruct the others ⇒ We suspect the Differential Privacy guarantees of Bagging [Liu et al., 2021] are in cause
- Solution time scales as $\Theta(N^2)$ with the number of examples, which is considerably better than the expected worst case ⇒ We can scale up the method to datasets including up to 1,500 examples

Table of Contents

1. Background
2. Illustrative Example: Reconstruction from Tree-Based Models
3. DRAFT: Dataset Reconstruction from Tree Ensembles
4. Experimental Study
5. **Discussion**

Summary

- Mathematical programming techniques permit us to build a new paradigm of data reconstruction attacks against machine learning models
- The reconstruction problem can be solved efficiently despite its theoretical hardness
- A trained Random Forest gives away most (if not all) of its training data, raising awareness on the dangers of releasing trained models

Future Works

- Using a similar strategy on **other machine learning models**
- **Improving the models** and including additional constraints modeling the behavior of the training algorithms
- **Developing and benchmarking protection measures** (in particular, differential privacy techniques) to avoid such attacks in the future

Contact & Links

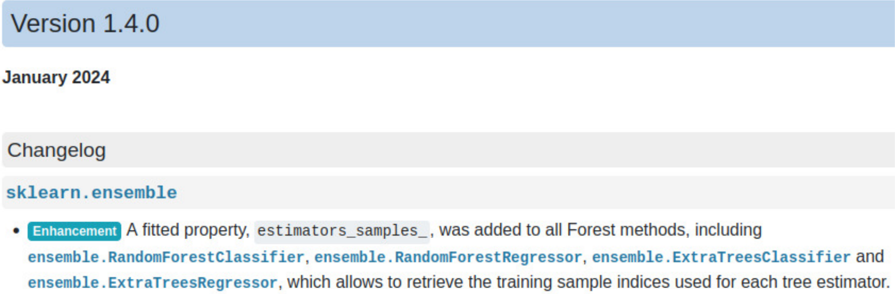
Contact

- Email: julien.ferry@polymtl.ca
- LinkedIn: https://www.linkedin.com/in/julien-ferry-9435341a7/

Paper:
https://openreview.net/pdf?id=cc72Vnfvoc

Code:
https://github.com/vidalt/DRAFT

Epilogue



Version 1.4.0

January 2024

Changelog

`sklearn.ensemble`

- Enhancement A fitted property, `estimators_samples_`, was added to all Forest methods, including `ensemble.RandomForestClassifier`, `ensemble.RandomForestRegressor`, `ensemble.ExtraTreesClassifier` and `ensemble.ExtraTreesRegressor`, which allows to retrieve the training sample indices used for each tree estimator.

References I

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. propublica (2016). *ProPublica, May*, 23.

Dinur, I. and Nissim, K. (2003). Revealing information while preserving privacy. In Neven, F., Beeri, C., and Milo, T., editors, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 202–210. ACM.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Ferry, J., Aïvodji, U., Gambs, S., Huguet, M.-J., and Siala, M. (2024). Probabilistic Dataset Reconstruction from Interpretable Models. In *2nd IEEE Conference on Secure and Trustworthy Machine Learning*, Toronto, Canada.

Gambs, S., Gmati, A., and Hurfin, M. (2012). Reconstruction attack through classifier analysis. In Cuppens-Boulahia, N., Cuppens, F., and García-Alfaro, J., editors, *Data and Applications Security and Privacy XXVI - 26th Annual IFIP WG 11.3 Conference, DBSec 2012, Paris, France, July 11-13,2012. Proceedings*, volume 7371 of *Lecture Notes in Computer Science*, pages 274–281. Springer.

Liu, H., Jia, J., and Gong, N. Z. (2021). On the intrinsic differential privacy of bagging. In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 2730–2736. ijcai.org.

References II

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Perron, L. and Didier, F. (2023). CP-SAT.

Yeh, I.-C. and hui Lien, C. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473–2480.

Reconstruction from a Decision Tree

Probabilistic Reconstructions

The knowledge acquired from a Decision Tree's structure:

- Can be encoded within a *probabilistic dataset*: a set of random variables (one for each attribute of each example)
- The reconstruction success can be quantified as the uncertainty reduction within the constructed *probabilistic dataset*
- Introduced by Gambs et al. [2012] and generalized in Ferry et al. [2024]

CP Formulation without Bagging I

CP Formulation without Bagging: Variables

We define variables: (i) modelling the reconstructed examples' attributes, and (ii) assigning the examples to the trees' leaves.

- $x_{ki} \in \{0;1\}$ is the value of feature i for example k in the reconstruction
- $y_{tvk} \in \{0;1\}$ indicates whether training example k is classified by leaf v within tree t

NB: Since each example appears exactly once within each tree's total counts, we can pre-assign classes to each of them. We let $z_{kc} \in \{0;1\}$ be 1 if training example k is part of class c, 0 otherwise.

CP Formulation without Bagging II

CP Formulation without Bagging: Constraints

- $\forall k \in \{1..N\}, \forall w \in vects : \sum_{i \in w} x_{ki} = 1$ One-Hot Encoding Consistency
- $\forall t \in \mathcal{T}, \forall k \in \{1..N\} : \sum_{v \in \mathcal{V}_t^L} y_{tvk} = 1$ Each example is assigned to exactly one leaf of each tree *(redundant)*
- $\forall t \in \mathcal{T}, \forall v \in \mathcal{V}_t^L : nb_{tvc} = \sum_{k \in \{1..N\}} z_{kc} y_{tvk}$ The counts match the number of assigned examples
- $\forall t \in \mathcal{T}, \forall k \in \{1..N\}, \forall v \in \mathcal{V}_t^L :$

  if y_{tvk} **then** $\left(\bigwedge_{i \in \Phi_v} x_{ki} = 1 \right) \wedge \left(\bigwedge_{i \in \overline{\Phi_v}} x_{ki} = 0 \right)$ The attributes' values are consistent with the splits

Results: Impact of d_{max} on the Reconstruction Time

Table 3: Average run time and number of runs for which the solver did not come up with a feasible solution (**#Runs**), for a fixed number of trees $|\mathcal{T}| = 100$ (default value).

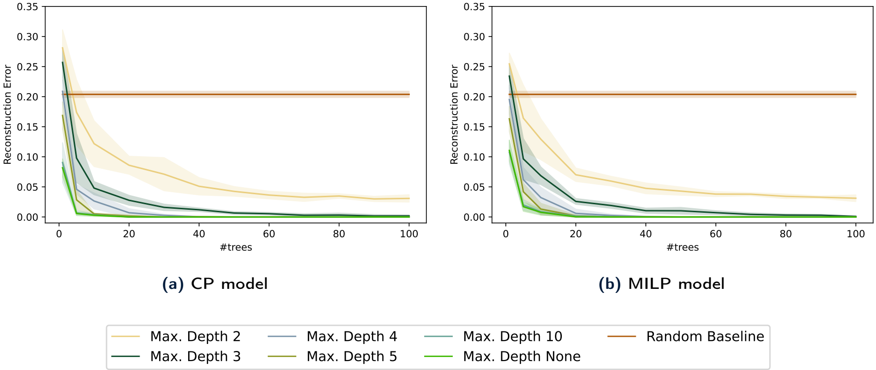
	Max.	No Bagging		Bagging
	Depth	Avg. T (s)	#Runs	#Runs
COMPAS	2	9.5	0	0
	3	36.5	0	0
	4	45.7	0	0
	5	70.0	0	0
	10	110.9	0	0
	None	100.8	0	0
Adult	2	20.2	0	0
	3	81.5	0	0
	4	1943.5	0	0
	5	1290.7	0	0
	10	346.4	0	1/5
	None	196.3	0	1/5
Default Credit	2	35.5	0	0
	3	300.8	1/5	0
	4	6040.0	2/5	0
	5	1358.5	0	0
	10	382.8	0	0
	None	165.0	0	4/5

Results: Impact of $|\mathcal{T}|$ on the Reconstruction Time

Table 4: Average run time and number of runs for which the solver did not come up with a feasible solution (**#Runs**), for no maximum depth constraint (default value).

	$\lvert\mathcal{T}\rvert$	No Bagging		Bagging
		Avg. T (s)	#Runs	#Runs
COMPAS	1	0.7	0	0
	10	8.4	0	0
	30	39.8	0	0
	50	53.0	0	0
	80	89.7	0	0
	100	100.8	0	0
Adult	1	0.8	0	0
	10	74.2	0	0
	30	84.9	0	0
	50	85.1	0	0
	80	119.9	0	0
	100	196.3	0	1/5
Default Credit	1	0.9	0	0
	10	129.7	0	0
	30	47.7	0	0
	50	66.2	0	2/5
	80	161.3	0	4/5
	100	165.0	0	4/5

MILP vs CP Formulation (Non-Bagging Case): Numerical Results I

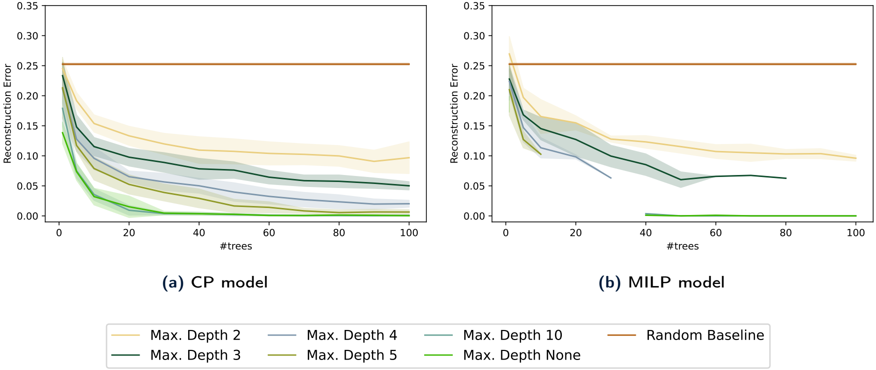


(a) CP model

(b) MILP model

Figure 7: Average reconstruction error as a function of the number of trees $|\mathcal{T}|$ within the attacked forest $\mathcal{T}$, for different maximum depth values d_{max} and for the random baseline. For the experiments on the COMPAS dataset, not using bagging, we report the results obtained using either the CP model or the MILP one.

MILP vs CP Formulation (Non-Bagging Case): Numerical Results II

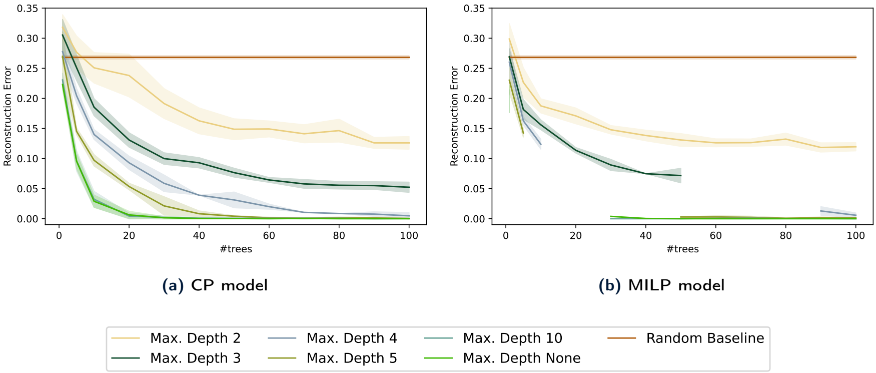


(a) CP model

(b) MILP model

Figure 8: Average reconstruction error as a function of the number of trees $|\mathcal{T}|$ within the attacked forest $\mathcal{T}$, for different maximum depth values d_{max} and for the random baseline. For the experiments on the Adult Income dataset, not using bagging, we report the results obtained using either the CP model or the MILP one.

MILP vs CP Formulation (Non-Bagging Case): Numerical Results III



(a) CP model

(b) MILP model

Figure 9: Average reconstruction error as a function of the number of trees $|\mathcal{T}|$ within the attacked forest $\mathcal{T}$, for different maximum depth values d_{max} and for the random baseline. For the experiments on the Default of Credit Card Client dataset, not using bagging, we report the results obtained using either the CP model or the MILP one.

MILP vs CP Formulation (Non-Bagging Case): Running Times

Table 5: Reconstruction times for the experiments on the COMPAS dataset.

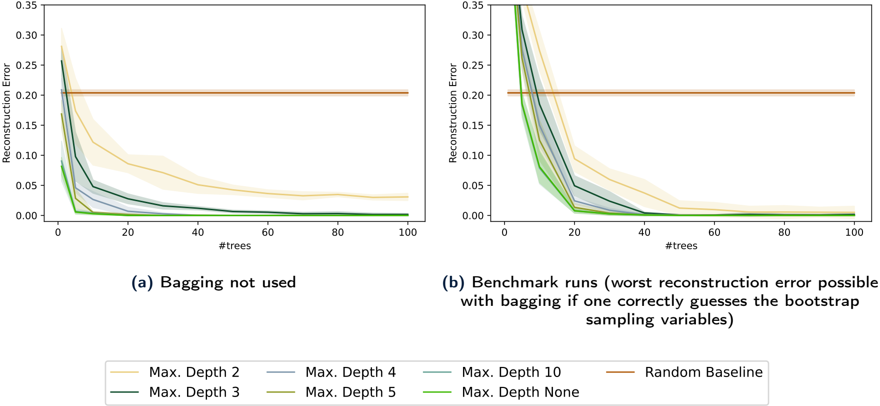
Max. Depth	Method	Reconstruction Times (s)			
		Avg	Std	Min	Max
2	CP	4.7	3.8	0.1	17.0
	MILP	5.4	6.1	0.1	31.2
3	CP	14.1	12.9	0.1	48.4
	MILP	10.3	21.9	0.2	162.5
4	CP	25.2	18.9	0.2	58.9
	MILP	24.7	52.6	0.2	302.7
5	CP	34.3	23.4	0.3	85.8
	MILP	26.4	60.4	0.3	418.6
10	CP	53.7	39.2	0.5	160.6
	MILP	77.6	312.7	2.6	2471.2
None	CP	49.7	35.5	0.6	142.0
	MILP	188.3	791.2	3.3	4521.8

Experiments on the Impact of Bagging

Setup

- Pre-fix the number of occurrences of each example to each tree
- Find out the **worst reconstruction possible** (*i.e.*, the one with the greatest error) among all those compatible with the constraints and the set assignment
- $\Longrightarrow$ we observe that if the number of occurences of each example within each tree's training set is correctly guessed, the reconstruction performance is as good (actually, better!) in the worst case as in the experiments without bagging
- $\Longrightarrow$ this illustrates the fact that the inherent difficulty of the reconstruction in the bagging case comes from the randomness added by Bagging (which is consistent with its Differential Privacy guarantees [Liu et al., 2021])

Experiments on the Impact of Bagging: Results I

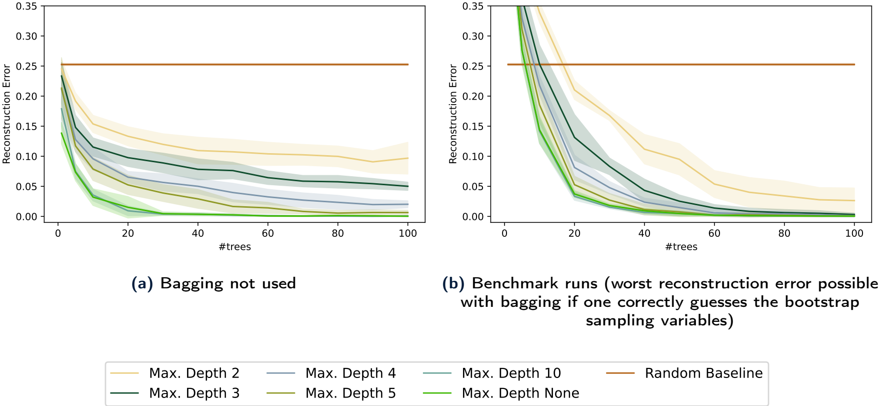


(a) Bagging not used

(b) Benchmark runs (worst reconstruction error possible with bagging if one correctly guesses the bootstrap sampling variables)

Figure 10: Comparison of the benchmark results (using bagging, worst possible reconstruction error using our set of constraints if the number of occurrences of each example within each tree are known) with the "no-bagging" ones, COMPAS dataset

Experiments on the Impact of Bagging: Results II

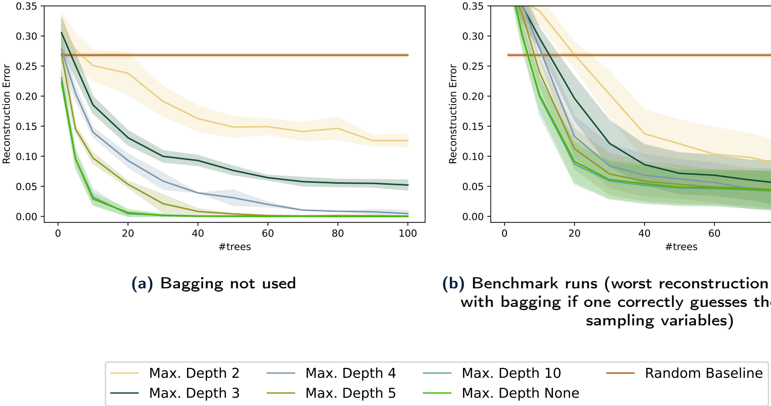


(a) Bagging not used

(b) Benchmark runs (worst reconstruction error possible with bagging if one correctly guesses the bootstrap sampling variables)

Figure 11: Comparison of the benchmark results (using bagging, worst possible reconstruction error using our set of constraints if the number of occurrences of each example within each tree are known) with the "no-bagging" ones, UCI Adult Income dataset

Experiments on the Impact of Bagging: Results III



(a) Bagging not used

(b) Benchmark runs (worst reconstruction error possible with bagging if one correctly guesses the bootstrap sampling variables)

Figure 12: Comparison of the benchmark results (using bagging, worst possible reconstruction error using our set of constraints if the number of occurrences of each example within each tree are known) with the "no-bagging" ones, Default of Credit Card Client dataset

Experiments on Scalability I

#Examples	RF Accuracy		Reconstruction error	Reconstruction Time (s)			
	Train	Test	Avg	Avg	Std	Min	Max
25	0.896	0.559	0.0	5.8	1.1	4.1	7.4
50	0.860	0.556	0.0	30.5	5.8	26.6	42.0
100	0.800	0.582	0.0	84.1	12.0	65.8	99.1
200	0.770	0.617	0.0	260.9	28.6	231.2	300.2
300	0.759	0.629	0.0	467.9	80.2	386.1	621.9
400	0.741	0.632	0.0	699.9	52.5	602.3	754.1
500	0.734	0.639	0.0	1071.9	197.0	883.9	1448.7
750	0.725	0.643	0.0	2219.4	428.4	1711.9	2818.9
1000	0.714	0.642	0.0	3678.8	231.0	3300.8	4005.0
1500	0.704	0.650	0.0	7362.2	1097.0	6485.9	9519.5

Table 6: Experiments (non-bagging case) on the reconstruction method's scalability, COMPAS dataset. Applying a simple power law regression, we observe that running times are in $\Theta(N^{1.7})$.

Experiments on Scalability II

#Examples	RF Accuracy		Reconstruction error	Reconstruction Time (s)			
	Train	Test	Avg	Avg	Std	Min	Max
25	0.952	0.709	0.0	9.1	2.9	6.2	14.6
50	0.964	0.748	0.0	37.0	4.4	29.2	42.8
100	0.964	0.770	0.0	188.2	58.0	117.4	278.3
200	0.949	0.767	0.0	631.7	110.8	513.2	838.6
300	0.935	0.771	0.1	4860.4	2457.7	1430.2	7695.8
400	0.925	0.778	0.1	6119.3	2365.6	2304.4	8054.8
500	0.913	0.779	0.1	6523.6	1558.4	4695.4	8429.2
750	0.905	0.780	0.0	13911.5	5146.7	8764.9	19058.2

Table 7: Experiments (non-bagging case) on the reconstruction method's scalability, UCI Adult Income dataset. Applying a simple power law regression, we observe that running times are in $\Theta(N^{1.4})$.

Experiments on Scalability III

#Examples	RF Accuracy		Reconstruction error	Reconstruction Time (s)			
	Train	Test	Avg	Avg	Std	Min	Max
25	0.968	0.733	0.0	6.7	1.5	4.0	8.1
50	0.976	0.723	0.0	42.1	3.5	37.8	47.3
100	0.972	0.740	0.0	148.3	16.6	127.6	169.4
200	0.965	0.751	0.0	905.6	528.1	584.1	1958.1
300	0.957	0.763	0.0	2369.8	1167.5	1316.7	4524.4
400	0.952	0.760	0.0	2733.2	581.5	2065.6	3706.5
500	0.947	0.765	0.0	6119.9	1880.1	3902.8	8671.4

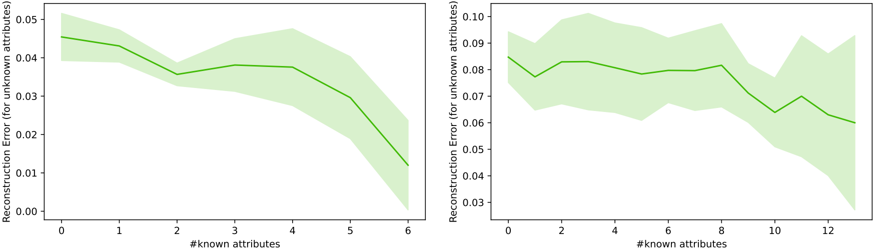
Table 8: Experiments (non-bagging case) on the reconstruction method's scalability, Default of Credit Card Client dataset. Applying a simple power law regression, we observe that running times are in $\Theta(N^{2.4})$.

Experiments on Partial Reconstruction

Setup

- Using Bagging
- Pre-fix a varying number of examples or attributes to their true value and observe the reconstruction error for the remaining ones
- Fixing a number of known examples
  - Does not really help reconstructing the remaining ones
  - $\implies$ we suspect the Differential Privacy guarantees of Bagging [Liu et al., 2021] are in cause
- Fixing a number of known attributes
  - Corresponds to the case where some attributes are public information
  - $\implies$ helps reconstructing the others!

Experiments on Partial Reconstruction: Results I

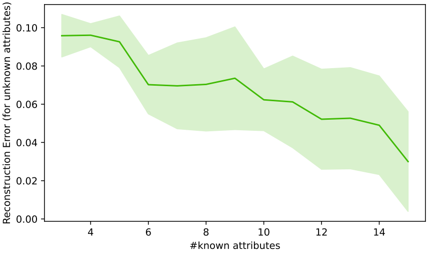


(a) COMPAS dataset

(b) UCI Adult Income dataset

Figure 13: Results of reconstruction experiments with knowledge of some of the attributes. We report the reconstruction error (for the unknown attributes) as a function of the number of known attributes in the forest's training set. For these experiments, all forests are learnt using `scikit-learn`'s default configuration (*i.e.*, $|\mathcal{T}| = 100$ and no maximum depth constraint). Reconstruction errors are averaged over 5 different random seeds and we also report the standard deviation.

Experiments on Partial Reconstruction: Results II



(a) Default of Credit Card Client dataset

Figure 14: Results of reconstruction experiments with knowledge of some of the attributes. We report the reconstruction error (for the unknown attributes) as a function of the number of known attributes in the forest's training set. For these experiments, all forests are learnt using `scikit-learn`'s default configuration (*i.e.*, $|\mathcal{T}| = 100$ and no maximum depth constraint). Reconstruction errors are averaged over 5 different random seeds and we also report the standard deviation.