Model Distillation

December 4, 2019

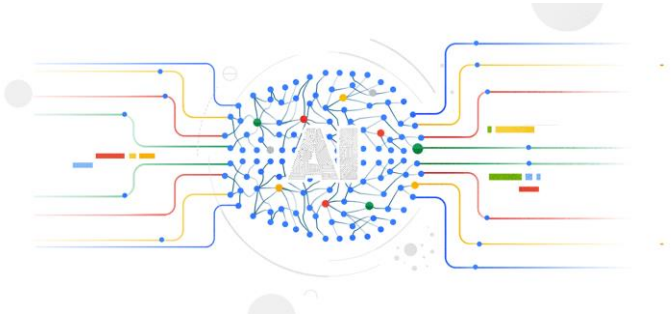
Google Cloud

Explaining AI

While Machine Learning models can identify correlations in data to achieve high accuracies when it comes to performing a myriad of tasks successfully, these models are still black boxes whose decisions cannot be trusted by simply looking at their structure or weights.

Understanding the behavior of a model is crucial in industries where confidence in the model is critical and interpretability is a must for the use of such models.

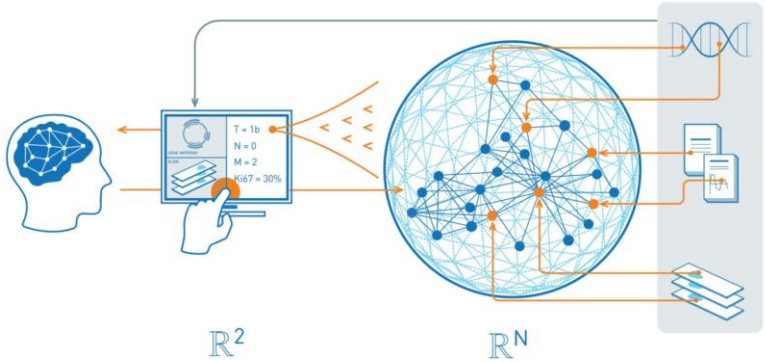
Explaining such models allows us to quantify the contribution of each of the model inputs to the model's output, thus enabling users to understand the decisions taken by the model. Furthermore, these explanations can be used to debug and improve the model and gain insights to be shared with the model's consumers.



Model Distillation

Explainability methods such as Integrated Gradients (IG) and SHapley Additive exPlanations (SHAP) simply attribute the decisions of neural networks to a subset of input features and allow us to rank feature importance.

Another axis of model explanations is to directly track the decision path or the “thought process” through the use of rules that represent the model, which is what Model Distillation allows us to do.



Google Cloud

Goal

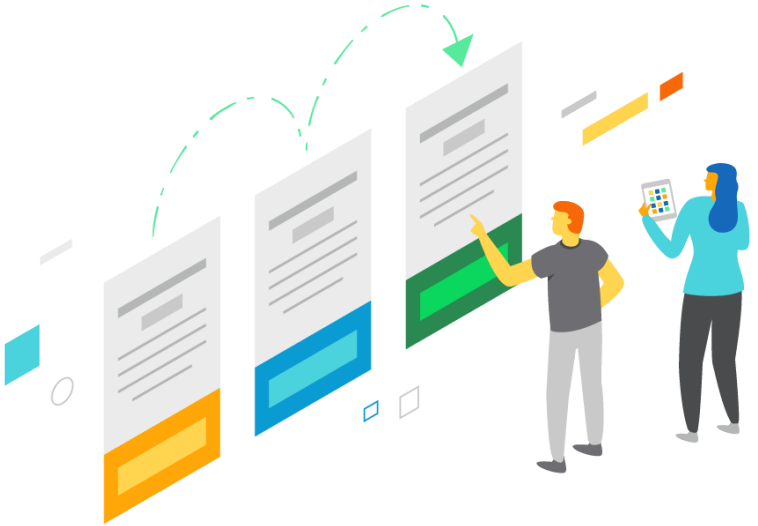
Extract rules using different Model Distillation techniques from models trained on different datasets and assess them to compare and contrast their effectiveness as Model Distillation techniques.

Model Distillation techniques considered:

- Sensei
- SuperSeti
- Soft Decision Trees
- Random Forests
- Gradient Boosted Decision Trees
- XGBoost Gradient Boosted Decision Trees

Datasets considered:

- Census
- Iris
- Mushroom
- Titanic



Google Cloud

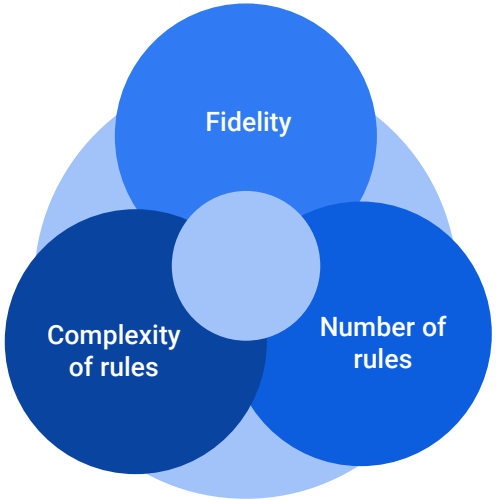
Trade-Offs

The three factors that are important when extracting rules from models are:

- Accuracy of the new model (Fidelity)
- Number of rules extracted
- Complexity (crosses) of rules extracted

Our goal is to minimize the number of rules and the complexity of rules extracted while maximizing the fidelity of the new model that the state-of-the-art model is distilled into.

There are trade-offs to optimizing these factors. Increasing fidelity increases the number and complexity of rules. Decreasing number of rules while maintaining the same fidelity increases the complexity of rules. Finally, decreasing complexity of rules while maintaining the same fidelity increases the number of rules.



Model Distillation Experiments and Results

Sensei

Sensei is a tool that is used to train of linear predictive models based on historical data. Essentially, Sensei builds a model of its own using the given distilled dataset and returns a set of rules that represent this new model. Sensei and SuperSeti are our baselines.

Advantages:

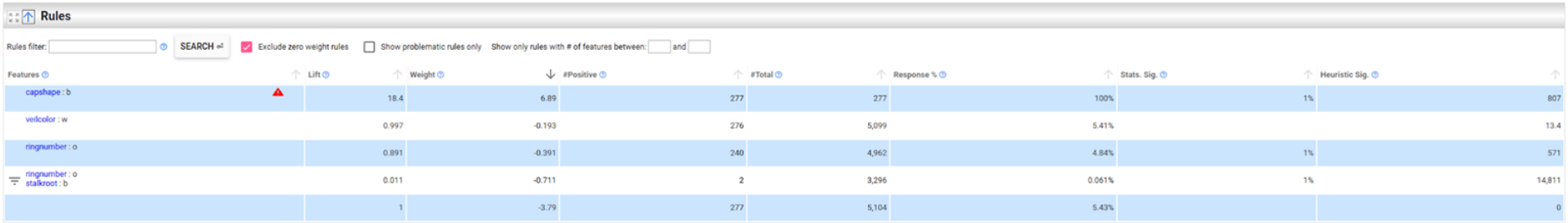
- Fewer rules than SuperSeti
- Less hyperparameter tuning

Disadvantages:

- More complex rules than SuperSeti
- Very slow
- Not as scalable as SuperSeti

Datasets	Fidelity	Number of rules	Complexity of rules
Census	0.922	400	5
Census Distilled	0.996	216	7
Iris	0.914	6	2
Iris Distilled	0.547	1	3
Mushroom	1.00	2	1
Mushroom Distilled	1.00	4	2
Titanic	0.852	5	2
Titanic Distilled	0.993	3	2

Sensei Output



Rules

Rules filter: SEARCH Exclude zero weight rules Show problematic rules only Show only rules with # of features between: and

Features	Lift	Weight	#Positive	#Total	Response %	Stats. Sig.	Heuristic Sig.
capshape : b	18.4	6.89	277	277	100%	1%	807
veilcolor : w	0.997	-0.193	276	5,099	5.41%		13.4
ringnumber : o	0.891	-0.391	240	4,962	4.84%	1%	571
ringnumber : o stalkroot : b	0.011	-0.711	2	3,296	0.061%	1%	14,811
	1	-3.79	277	5,104	5.43%		0

Google Cloud

SuperSeti

SuperSeti is a large-scale machine learning system that is good at building models for high-dimensional, sparse feature spaces in addition to automatically exploring and creating feature crosses. SuperSeti functions the same way Sensei does to return a set of rules that represent the distilled model. Sensei and SuperSeti are our baselines.

Advantages:

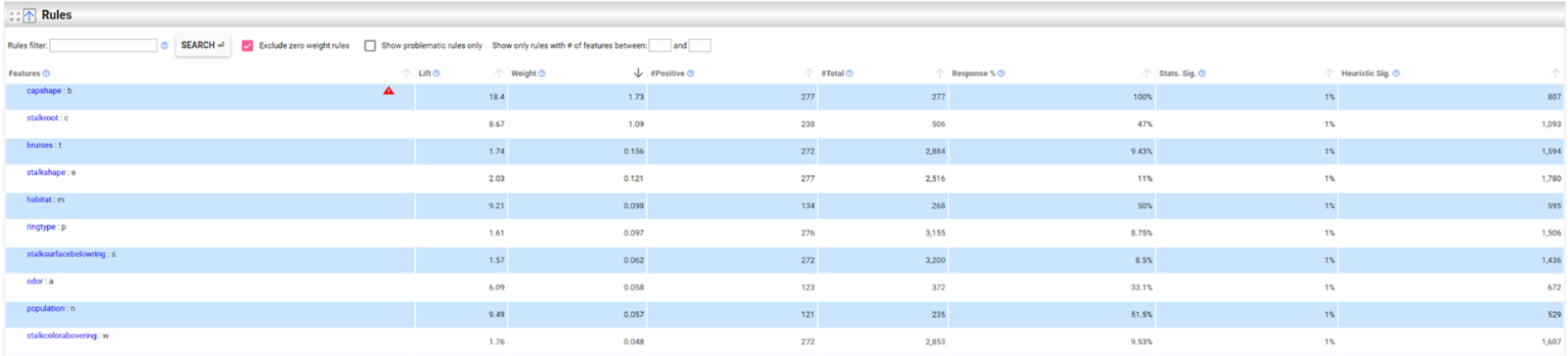
- Highly scalable
- Less complex rules than Sensei
- Less hyperparameter tuning

Disadvantages:

- Large number of rules
- Very slow
- Can only be used for binary classification problems (Ex: Cannot be used with Iris dataset)

Datasets	Fidelity	Number of rules	Complexity of rules
Census	0.927	243	3
Census Distilled	0.996	293	3
Iris	-	-	-
Iris Distilled	-	-	-
Mushroom	1.00	82	2
Mushroom Distilled	1.00	42	1
Titanic	0.64	10	1
Titanic Distilled	0.996	13	1

SuperSeti Output



Rules

Rules filter: SEARCH Exclude zero weight rules Show problematic rules only Show only rules with # of features between: and

Features	Lift	Weight	#Positive	#Total	Response %	Stats. Sig.	Heuristic Sig.
capshape : b	18.4	1.73	277	277	100%	1%	807
stalkroot : c	8.67	1.09	238	506	47%	1%	1,093
bruises : t	1.74	0.156	272	2,884	9.43%	1%	1,594
stalkshape : e	2.03	0.121	277	2,516	11%	1%	1,780
habitat : m	9.21	0.098	134	268	50%	1%	595
ringtype : p	1.61	0.097	276	3,155	8.75%	1%	1,506
stalksurfacebelowring : s	1.57	0.062	272	3,200	8.5%	1%	1,436
odor : a	6.09	0.058	123	372	33.1%	1%	672
population : n	9.49	0.057	121	235	51.5%	1%	529
stalkcolorabovering : w	1.76	0.048	272	2,853	9.53%	1%	1,607

Google Cloud

Soft Decision Trees

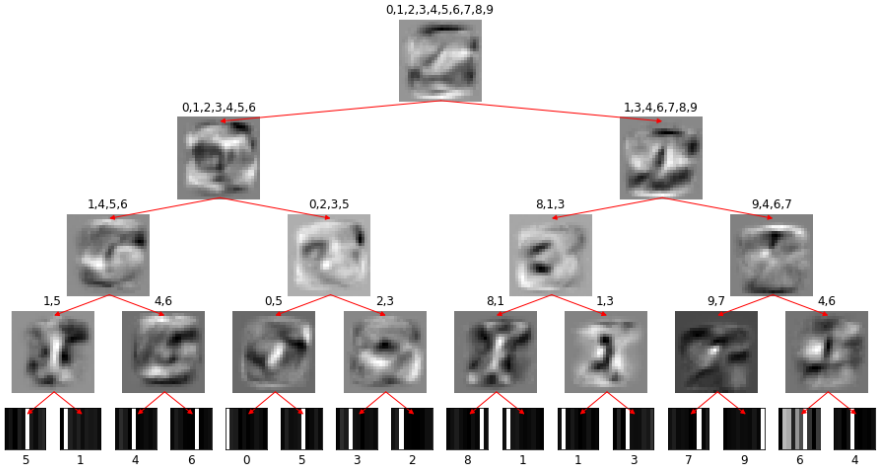
Soft Decision Trees (SDTs) are Decision Trees with a softening function (This allows the Soft Decision Tree to generalize better) applied at each of the nodes such that both children are chosen with probabilities given by a sigmoid gating function.

Advantages:

- Fewer rules than baselines
- Less complex rules than baselines

Disadvantages:

- Slower (Slowest among all techniques) than baselines
- Not as scalable as SuperSeti
- Requires a lot of hyperparameter tuning



Google Cloud

Random Forests

Random Forests (RFs) are meta estimators that fit a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A Random Forest is trained on the distilled dataset and rules are extracted from each of the trees.

Advantages:

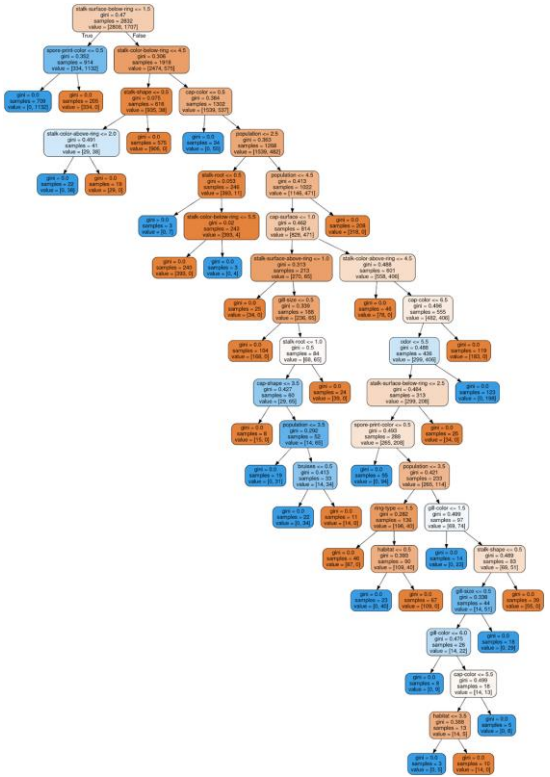
- Higher fidelity than baselines on smaller datasets
- Faster than baselines

Disadvantages:

- Lower fidelity than baselines on larger datasets
- Large number of rules than baselines
- More complex rules than baselines
- Not as scalable as SuperSeti
- Requires a lot of hyperparameter tuning
- Similar trees created since tree creation is random

Datasets	Fidelity	Number of rules	Complexity of rules
Census	0.82	13289	4
Census Distilled	0.89	8226	4
Iris	0.65	469	4
Iris Distilled	0.85	76	4
Mushroom	0.999	270	11
Mushroom Distilled	1.00	205	12
Titanic	0.78	1262	7
Titanic Distilled	0.99	273	7

Random Forest Output



1: if cap-surface < 0.5 then poisonous
2: if 0.5 <= cap-surface if stalk-color-below-ring < 4.5 if cap-shape < 1.0 then edible
3: if 0.5 <= cap-surface if stalk-color-below-ring < 4.5 if 1.0 <= cap-shape then poisonous
4: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if gill-color < 0.5 if cap-shape < 1.0 then edible
5: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if gill-color < 0.5 if 1.0 <= cap-shape then poisonous
6: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color if cap-color < 5.5 if ring-number < 1.5 then poisonous
7: if 0.5 <= cap-surface < 2.5 if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if cap-color < 5.5 if 1.5 <= ring-number if habitat < 2.0 if cap-shape < 1.0 then edible
8: if 0.5 <= cap-surface < 2.5 if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if cap-color < 5.5 if 1.5 <= ring-number if habitat < 2.0 if 1.0 <= cap-shape then poisonous
9: if 2.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if cap-color < 2.5 if 1.5 <= ring-number if habitat < 2.0 if cap-shape < 1.0 then edible
10: if 2.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if cap-color < 2.5 if 1.5 <= ring-number if habitat < 2.0 if 1.0 <= cap-shape then poisonous
11: if 2.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if 2.5 <= cap-color < 5.5 if 1.5 <= ring-number if habitat < 2.0 then edible
12: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if cap-color < 2.5 if 1.5 <= ring-number if 2.0 <= habitat if cap-shape < 1.0 then edible
13: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if cap-color < 2.5 if 1.5 <= ring-number if 2.0 <= habitat if 1.0 <= cap-shape then poisonous
14: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 6.0 if 2.5 <= cap-color < 5.5 if 1.5 <= ring-number if 2.0 <= habitat then poisonous
15: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 6.0 <= gill-color if cap-color < 2.5 if 1.5 <= ring-number if habitat < 2.0 then poisonous
16: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 6.0 <= gill-color if 2.5 <= cap-color < 5.5 if 1.5 <= ring-number if habitat < 2.0 if cap-shape < 1.0 then edible
17: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 6.0 <= gill-color if 2.5 <= cap-color < 5.5 if 1.5 <= ring-number if habitat < 2.0 if 1.0 <= cap-shape then poisonous
18: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 6.0 <= gill-color if cap-color < 2.5 if 1.5 <= ring-number if 2.0 <= habitat < 3.5 if cap-shape < 1.0 then edible
19: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 6.0 <= gill-color if cap-color < 2.5 if 1.5 <= ring-number if 2.0 <= habitat < 3.5 if 1.0 <= cap-shape then poisonous
20: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 6.0 <= gill-color if 2.5 <= cap-color < 5.5 if 1.5 <= ring-number if 2.0 <= habitat < 3.5 then poisonous
21: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 6.0 <= gill-color if cap-color < 5.5 if 1.5 <= ring-number if 3.5 <= habitat then poisonous
22: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color if 5.5 <= cap-color if stalk-root < 0.5 if cap-shape < 1.0 then edible
23: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color if 5.5 <= cap-color if stalk-root < 0.5 if 1.0 <= cap-shape then poisonous
24: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color if 5.5 <= cap-color if 0.5 <= stalk-root < 1.5 if gill-spacing < 0.5 if cap-shape < 2.5 then edible
25: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color if 5.5 <= cap-color if 0.5 <= stalk-root < 1.5 if gill-spacing < 0.5 if 2.5 <= cap-shape then poisonous
26: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 7.5 if 5.5 <= cap-color if 0.5 <= stalk-root < 1.5 if 0.5 <= gill-spacing if cap-shape < 1.0 then edible
27: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color < 7.5 if 5.5 <= cap-color if 0.5 <= stalk-root < 1.5 if 0.5 <= gill-spacing if 1.0 <= cap-shape then poisonous
28: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 7.5 <= gill-color if 5.5 <= cap-color if 0.5 <= stalk-root < 1.5 if 0.5 <= gill-spacing then poisonous
29: if 0.5 <= cap-surface if 4.5 <= stalk-color-below-ring if 0.5 <= gill-color if 5.5 <= cap-color if 1.5 <= stalk-root then poisonous

Google Cloud

Gradient Boosted Decision Trees

Gradient Boosted Decision Trees (GBDTs) build an additive model in a forward stage-wise fashion that allows for the optimization of arbitrary differentiable loss functions. They are similar to Random Forests but use previous trees for the creation of the next trees and thus, the subsequent trees created are not “random”.

Advantages:

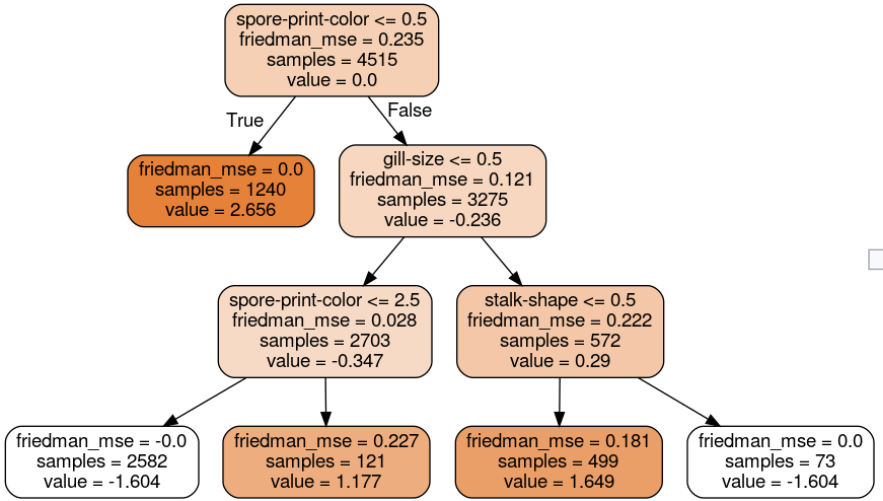
- Higher fidelity than baselines on smaller datasets
- Faster than baselines
- Fewer rules than baselines
- Less complex rules than baselines

Disadvantages:

- Lower fidelity than baselines on larger datasets
- Not as scalable as SuperSeti
- Requires a lot of hyperparameter tuning

Datasets	Fidelity	Number of rules	Complexity of rules
Census	0.828	80	3
Census Distilled	0.902	66	3
Iris	0.929	64	3
Iris Distilled	0.964	46	2
Mushroom	0.974	50	3
Mushroom Distilled	1.00	20	1
Titanic	0.846	80	3
Titanic Distilled	1.00	40	3

Gradient Boosted Decision Tree Output



1: if spore-print-color < 0.5 then poisonous
2: if 0.5 <= spore-print-color < 2.5 if gill-size < 0.5 then edible
3: if 2.5 <= spore-print-color if gill-size < 0.5 then poisonous
4: if 0.5 <= spore-print-color if 0.5 <= gill-size if 0.5 <= stalk-shape then poisonous
4: if 0.5 <= spore-print-color if 0.5 <= gill-size if stalk-shape < 0.5 then edible

Google Cloud

XGBoost Gradient Boosted Decision Trees

XGBoost Gradient Boosted Decision Trees (XGBTs) are similar to normal GBDTs except for the fact that XGBTs perform gradient boosting from one level of the tree to the next and thus, result in fewer rules that are less complex.

Advantages:

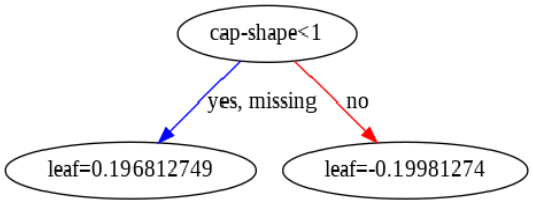
- Higher fidelity than baselines on smaller datasets
- Faster (Fastest among all techniques) than baselines
- Fewer (Fewest among all techniques) number of rules than baselines
- Less complex rules compared to baselines
- Highly scalable like SuperSeti

Disadvantages:

- Lower fidelity than baselines on larger datasets
- Requires some (not a lot) hyperparameter tuning

Datasets	Fidelity	Number of rules	Complexity of rules
Census	0.832	35	4
Census Distilled	0.902	22	4
Iris	0.852	32	4
Iris Distilled	0.926	10	2
Mushroom	1.00	10	6
Mushroom Distilled	1.00	2	1
Titanic	0.783	18	5
Titanic Distilled	1.00	4	3

XGBoost Gradient Boosted Decision Tree Output



1: if cap-shape < 1 then edible
2: if 1 <= cap-shape then poisonous

Comparison of the Model Distillation Results

The results (Fidelity / Number of rules / Complexity of rules) for the Model Distillation techniques considered are:

Distilled Dataset	Sensei			SuperSeti			RFs			GBDTs			XGBTs		
Census	0.996	216	7	0.996	293	3	0.89	8226	4	0.902	66	3	0.902	22	4
Iris	0.547	1	3	-	-	-	0.85	76	4	0.964	46	2	0.926	10	2
Mushroom	1.00	4	2	1.00	42	2	1.00	205	12	1.00	20	1	1.00	2	1
Titanic	0.993	3	2	0.996	13	1	0.99	273	7	1.00	40	3	1.00	4	3

Comparative Analysis

Criteria		Sensei	SuperSeti	SDTs	RFs	GBDTs	XGBTs	Additional Information
1	High fidelity on smaller datasets	✗	✗	✓	✓	✓	✓	• XGBTs are the best • Smaller datasets do not require a lot of feature exploration
2	High fidelity on larger datasets	✓	✓	✗	✗	✗	✗	• Sensei is the best • Larger datasets require a lot of feature exploration
3	Low number of rules generated	✗	✗	✓	✗	✓	✓	• XGBTs are the best
4	Low complexity of rules generated	✗	✓	✓	✗	✓	✓	• XGBTs are the best
5	Low time taken for rule generation	✗	✗	✗	✓	✓	✓	• XGBTs are the best
6	High scalability	✗	✓	✗	✗	✗	✓	• XGBTs and SuperSeti are the best

Google Cloud

Moving Forward with XGBTs

The XGBT hyperparameters that directly impact the performance of the distilled model are:

- max_depth - Maximum tree depth for base learners.
- learning_rate - Boosting learning rate (xgb's "eta").
- n_estimators - Number of trees to fit.
- gamma - Minimum loss reduction required to make a further partition on a leaf node of the tree.
- min_child_weight - Minimum sum of instance weight (hessian) needed in a child.
- max_delta_step - Maximum delta step we allow each tree's weight estimation to be.
- reg_alpha - L1 regularization term on weights.
- reg_lambda - L2 regularization term on weights.

Tuning these hyperparameters will allow us to increase fidelity while generating a low number of rules with low complexities. Perhaps these hyperparameters can be chosen by users based on their needs.

Alternatives and Improvements Considered

Alternatives:

- TensorFlow Gradient Boosted Decision Trees (TFBTs) - They are similar to XGBTs in the sense that they have gradient boosting within each level of the tree and are highly scalable. However, TFBTs are much slower compared to XGBTs and require a lot of pre-processing of the given dataset for training and a lot of post-processing of the trees for rule extraction, which makes XGBTs the better alternative.
- Rotation Forests - They are similar to Random Forests but use feature extraction techniques such as PCA to reduce the dimensionality of the input to obtain a lower number of rules with lower complexities. However, the rules extracted will not be interpretable in the original input dimensions, which makes Rotation Forests unviable.

Improvements:

- Tree pruning - The CART algorithm, tree pruning algorithms, and early stopping techniques reduce overfitting and are worth looking into because they reduce the number and complexity of the rules generated.
- Feature exploration - While feature exploration techniques are slow, they would allow decision trees to perform better on larger and more complex datasets.

Challenges Faced

Some of the major engineering challenges that were tackled:

Soft Decision Trees	Soft Decision Trees are not a part of any existing Machine Learning library available to the public	Implemented the Soft Decision Tree classifier for Model Distillation based on the academic paper from scratch
Rule Scoring Metrics	No metrics to compare the rule generation capabilities of different Model Distillation techniques	Created several rule scoring metrics to compare the Model Distillation Techniques, some of which evaluate the techniques based the user's preferences
Distillation Trade-offs	How to proceed forward given the results from the previous experiments	Analyzed the Model Distillation Trade-offs using the rule scoring metrics to minimize the number and complexity of the generated rules while maximizing the fidelity of the distilled model generation capabilities
Feature Exploration	Decision Tree based Model Distillation techniques do not perform feature exploration	Used existing feature exploration methods to explore the feature space and generate feature crosses that are then fed to the Decision Tree based Model Distillation techniques
Rule Extraction	Machine Learning libraries do not have modules for extracting rules from the Decision Trees	Created rule extraction modules for different libraries that extract rules from Decision Trees and output rules in the same format as the baselines to compare their rule

What I've learned

- The necessity and impact of Explainable AI and its urgency in the industry right now
- Organizing my work and setting reasonable goals to tackle in the foreseeable future
- The importance of writing clean and concise code that is readable by everyone
- Using academic literature and internal resources to figure out the next best approach
- Scoping and breaking down a big project into smaller and more manageable components
- Documenting my work and ensuring that the results are reproducible for the next person to follow my progress

Google Cloud

Questions?

Thank you @xai-dev!