Decision tree leaf nodes as a tool for interpretability

Benoît Paris

benoit@explicable.ml

October 1, 2018

Abstract

Here is our submission paper for the explainable.ml FICO challenge, which is evaluated on the quality of global and local interpretations. Using the participation of an instance in a decision tree leaf node as a signal, we provide: a global view of the dataset that separates the instances according to the way their characteristics are related to the prediction. We also provide a more formal, simplicity-focused global interpretation. Using a Kolmogorov-like complexity definition, we show we can improve on existing techniques without compromising the model performance. In addition, we provide local interpretations linked to this global interpretation, providing a coherent view across the local and global interpretation scales.

Dataset & Task

The Dataset at hand is composed of 10459 instances, and is based around real-life data of home equity line credits. The features represent characteristics of a potential credit recipient, such as the number of recent late payments, or an external risk estimate. The Task is to build a model to predict if the credit will incur a payment default event.

Such tasks are very common in sectors such as banking and insurance. These sectors are heavily regulated so as to maintain useful and enforceable properties in the client-business-regulator relationship. Businesses are required to disclose what data was used, and what was the influence of the features over the prediction outcome. This helps in enforcing guarantees about fair treatment; it is an integral part of a transparency process.

Straightforward linkage between input and output can be easily obtained with using linear models; and these can often be the default option in a sector requiring interpretability. However, these linear models leave performance on the table; with regards to non-linear contributions and feature interactions, for example. This is why new approaches that remain interpretable yet address this performance void are needed.

Criteria

There have been several definitions of what constitutes a good interpretation. Here is the framework of criteria under which we will operate, and how they apply to our task:

- Society-compatible: An interpretation should be usable by all relevant actors, and they should be able to talk discuss it between them. In our loan default task, the different actors would be the data scientists, high-level executives, banking operators, credit line recipents, regulators.
- Simple, sparse model: physical phenomena tend to converge to laws with simple expressions. It should be a goal for a model to converge to simple interpretations, as it might

mirror better the real-life processes at hand. Also, the previously mentioned actors would cognitively benefit from simple interpretations as well.

- Feature complexity: The base items used for explanations should be simple. The model sparsity constraint should not be satisfied by hiding complexity into feature engineering.
- Monotonic: A feature increase/decrease should go in hand monotonically with its effect over the target outcome. This should help in the cognitive load as well.

Our approach

We take the approach of a using one simple concept and building interpretations on it. We use rule conditions. They are defined by the following:

- A variable
- A comparator (> or <=)
- A threshold

We posit that rule conditions are society-compatible. They are simple, clear in their definition, they can support communication in a simple fashion between the widest audiences.

A rule condition remaining valid for a range of instances satisfies the monotonicity goal as well. As the variable evolves on one side of the threshold, an effect attached to a rule condition being valid is to be chosen constant; and monotonicity does not require strict increase/decrease.

A rule is defined by a set of rule conditions, and a rule is valid if all its rule conditions are valid. A branch going from root node to leaf node in a decision tree can be thought of as a rule. Rules that make use of large number of rule conditions can thus affect our feature complexity goal.

Decisions trees build on rule conditions as their base element. Algorithms such as Random Forests[1] make intensive use of decision trees to obtain well-performing models. Variants like XGBoost[2] often yield the best models in machine learning competitions.

The number of rule conditions in decision trees can get large. Also, in order to obtain good model performance in Random Forests and XGBoost, large sets of decision trees need to be built. If we are to use rules extracted from decision trees, the number of rules getting high can affect our model-sparsity goal.

Global intuitive interpretation

Decision trees build their rules by splitting recursively the dataset, at each split crafting the rule condition that will best separate the instances over the label. In this regard it is a form of feature engineering. Applying these rules to each instance, we obtain a leaf node participation space. This high-dimensional space tries to separate best the instances over the output label. It also separates the instances according to the different ways one output label can be detected. Feature interactions and non-linear components that trees produce are present in this space as well.

[1] Leo Breiman - 2001 - https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
[2] Tianqi Chen, Carlos Guestrin - 2016 - https://arxiv.org/abs/1603.02754

Recent tools such as t-SNE[3] or UMAP[4] help reduce high-dimensional spaces to plottable 2D or 3D spaces. Interpretable linkage between inputs and outputs is lost, though, as the final dimensions created are quite remote from the original features. We use these tools to visualize the high-dimensional space of decision tree node participation.

In the challenge we do not recommend using such a plot as a final interpretable answer. Discussing the merits of local neighborhoods in the embedded space does not stand to robust and stable review, and thus is not society-compatible. Also, the 2 or 3 projected dimensions are opaque and highly complex in the linkage between the input feature and the output label.

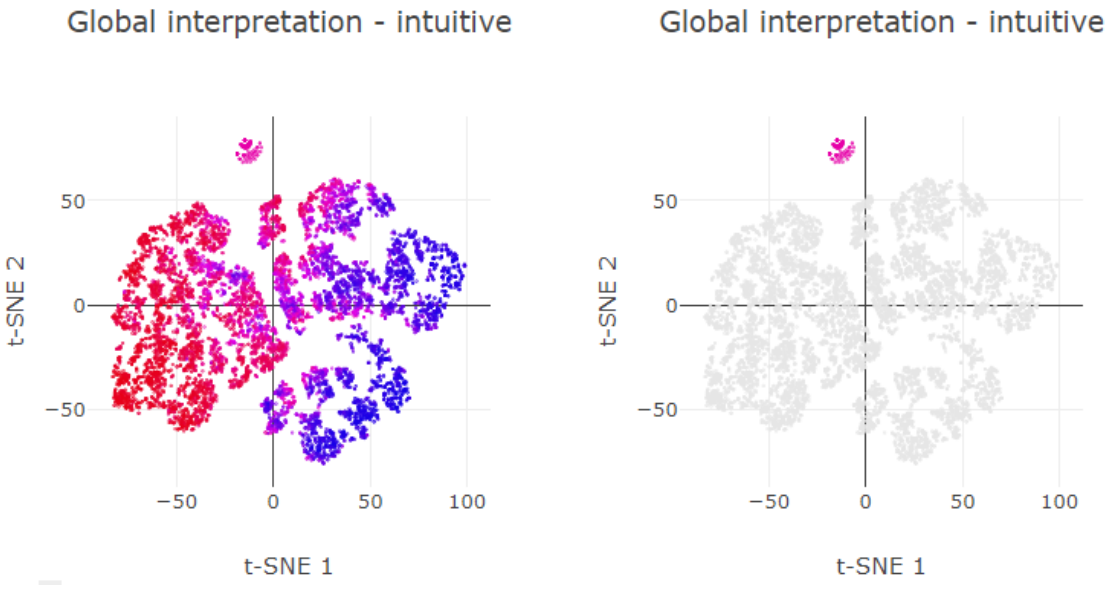
But the final dimensions being low displays a remarkable model sparsity, if we are to build a model to segment it. As a parallel, selecting spots of interest is what humans do when inspecting the surface of an object for interesting patterns.

We do recommend use of these plots by data-scientists as a global interpretation tool; for identifying and building more formal features that might be relevant and have an intrinsic meaning that stand review by society.

As part of our submission to the challenge, we provide a notebook to generate such a plot ("Global interpretation - intuitive.ipynb"). We evaluate this graphical representation but using the 2D coordinates as input features; with a 0.8-0.2 train-test split we obtain an AUC of around 0.78, which is not far from an optimal AUC.

The instances are colored according to their predicted outcome, blue for low probability of default, red for a high probability. We set the saturation with a feature that we want to highlight. This shows if the algorithm deemed it important to set aside a value as a special case to be handled differently. Below are two views; the first with no highlight, the second with values of MSinceMostRecentInqexcl7days at -9. The FAQ of the challenge tells us this value indicates that an investigation of a credit bureau was not needed as the applicant is probably a VIP.

In the simplicity focused explanation, we find that special treatment is indeed applied.



[3] L.J.P. van der Maaten and G.E. Hinton - 2008 - https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf
[4] Leland McInnes, John Healy - 2018 - https://github.com/lmcinnes/umap

Global simplicity-focused, society-compatible interpretation

As mentioned previously, rule condition usage has society-compatible and monotonic properties; but suffer from low model sparsity, and might represent complex features if taken out of deep trees.

Approaches such as RuleFit[5] work by building decision tree forests, extract the rules, and then use these rules as features to be fitted in a linear model. In this regard, it is a bridge between the classical linear methods that would be augmented by new features that deal with feature interactions and non-linearities.

Various strategies are employed to select rules among the many created by the trees. Pruning a lot of rules helps in having a sparse model, but does nothing for having low-complexity ones and be feature-simple.

Using the same approach, we introduce a custom measurement of rule complexity. We use and modify an implementation of RuleFit; in which we introduce weights for rules. Also, rules can be used in tandem with linear features, which could help in transitioning from purely linear models used in the industry to more convoluted rules.

In Rulefit, an indicator matrix is used to signal whether an instance falls in a leaf:

$$I_{RuleFit_{ij}} = \begin{cases} 1 & \text{if } X_i \text{ falls in leaf } N_j \\ 0 & \text{otherwise} \end{cases}$$

For i = 1..n, n being the number of instances; j = 1..q, q being the number of leaf nodes; and the Xi are the input vectors.

A Lasso regularization is then applied for selecting leaf-node participations:

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - I\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

Where y is the outcome, Beta is a vector of size q, and lambda a regularization parameter. The Lasso process tries to minimize the Beta complexity vector by picking features to use (which here are our leaf node participations) in a coordinate descent fashion. The leaf nodes that are not used will stay at zero in Beta.

The Beta term is constrained by the L1 norm, which is to say that leaf nodes are of equal importance with regards to the selection process; and will only be selected by the amount of information they contain.

For the purposes of controlling feature complexity, we introduce weights in the indicator matrix, so that the regularization will favor terms according to a custom importance.

We choose to penalize leaf nodes according to the number of rules conditions that compose them. Eg 'A > 1' is penalized by 1, when 'A > 1 & B > 1' is penalized by 2.

[5] Jerome H. Friedman and Bogdan E. Popescu - 2005 - http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf

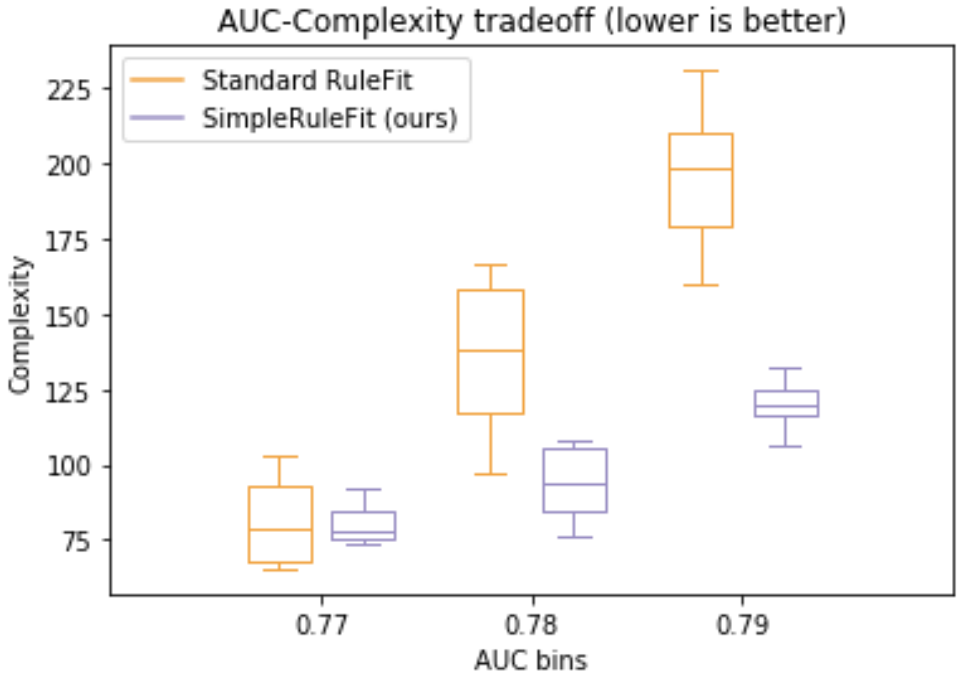
The indicator matrix becomes:

$$I_{SimpleRuleFit_{ij}} = \begin{cases} \frac{1}{w_j} & \text{if } X_i \text{ falls in leaf } N_j \\ 0 & \text{otherwise} \end{cases}$$

Where w_j is the penalty that we attach to rule j.

Considering two rules containing the same amount of information, one having 2 rules conditions, the second having 1 condition; fitting Beta will require twice the L1 'capital' for the former compared to the latter.

We evaluate these two versions of RuleFit using a range of constraints in the number of rules; and compare the performance-complexity tradeoffs between RuleFit and our version (using the challenge dataset, no feature engineering, rules-only mode, AUC as the performance measurement, 5-fold cross validation):



Complexity is the number of overall rule conditions used.

As part of our submission to the challenge we provide:

- An implementation of this custom RuleFit (named "CustomRuleFit", with installation instructions in README.md)
- A notebook which evaluates the tradeoffs ("RuleFitCustom – Evaluation.ipynb")
- A list of rules, obtained with a run of this algorithm on the challenge data ("RuleFitCustom - Global and Local explanations.ipynb")

Such a description of a model tends to optimize to a Kolmogorov-like complexity, as our goal is to lower the processing required in a model and get closer to the minimal description length.

The complexity here is the number of rule conditions as our base element of description. In practice most rules we obtain are composed of single rule conditions among a reduced set of features. These

rules can be grouped together by the variable used, and we find that most of them display a monotonic behavior.

Local interpretations

For the local interpretation we use feature attributions to display influence from input features to the output label. We find that it helps in creating concise, formal explanations suitable for communication. E.g. “You have variable X at Y, which augments the default risk score by Z, which is above our threshold of authorizing the credit line”.

Recent techniques such as TreeInterpreter[6] and Tree SHAP[7] have been developed to transform forests of decision trees into feature attributions. Specifically, Tree SHAP is an improvement on TreeInterpreter, in that it renders it stable by using Shapley values of feature influences. Shortly described, Shapley values are the average marginal contributions of a feature over all possible coalitions of features. Decision trees have a greedy behavior, focusing locally on the best split; and thus, are prone to skewed feature attributions. Shapley values offset this greedy behavior.

Approaches such as these help a lot in providing robust, stable attributions; and answer the question: if I didn’t have this feature, how well would I have been able to make the prediction? In other terms: were I to lack information on some features, how well having the other feature would help?

In the challenge, we did choose a very greedy strategy in selecting the rules, in building our global explanation. Our goal was to minimize the number of rule conditions. In this process, we did eliminate the usage of a few features as well.

We do want to retain a linkage between global and local explanations. This linkage is key in the client-business-regulator relationship. With this linkage a client can come towards the regulator with an explanation that was given by the business. The regulator and business can entertain a discussion on an existing and formal element. And the business can then correct its rule selection process to modify globally its relation with the clients.

This is why we will not present local feature attributions that display robustness to feature absence in their construction; our greediness towards simplicity prevents us from doing it. We present feature attributions built with the coefficients in the linear model. These feature attributions are calculated using the regular indicator matrix and applying the relevant coefficients. We then aggregate them on rules that are made of single rule conditions.

The attributions is available as a function in “RuleFitCustom - Global and Local explanations.ipynb”.

Conclusion

We have provided two explanatory models. One that is intuitive and to be used for intuitive explanatory data analysis. Another one, that can be used in a legislated setting in the client-business-regulator relationship. It has been built with simplicity and clarity constraints, as well as a local-global linkage consistency constraint.

---

[6] Ando Saabas - 2014 - http://blog.datadive.net/interpreting-random-forests/

[7] Scott M. Lundberg, Gabriel G. Erion, Su-In Lee - 2018 - https://github.com/slundberg/shap