Summary

imodels is a Python package for concise, transparent, and accurate predictive modeling. It provides users a simple interface for fitting and using state-of-the-art interpretable models, all compatible with scikit-learn (Pedregosa et al., 2011). These models can often replace black-box models while improving interpretability and computational efficiency, all without sacrificing predictive accuracy. In addition, the package provides a framework for developing custom tools and rule-based models for interpretability.

Statement of need

Recent advancements in machine learning have led to increasingly complex predictive models, often at the cost of interpretability. There is often a need for models which are inherently interpretable (Murdoch et al., 2019; Rudin, 2019), particularly in high-stakes applications such as medicine, biology, and political science. In these cases, interpretability can ensure that models behave reasonably, identify when models will make errors, and make the models more trusted by domain experts. Moreover, interpretable models tend to be much more computationally efficient than larger black-box models.

imodels aims to fill this gap by providing a simple unified interface and implementation for many state-of-the-art interpretable modeling techniques.

Features

Interpretable models can take various forms. Figure 1 shows four possible forms a model in the imodels package can take. Each form constrains the final model in order to make it interpretable, but there are different methods for fitting the model which differ in their biases and computational costs. The imodels package contains implementations of various such methods and also useful functions for recombining and extending them.

Rule sets consist of a set of rules which each act independently. There are different strategies for deriving a rule set, such as Skope-rules (Skope Collaboration, 2021) or Rulefit (Friedman et al., 2003).
Rule lists are composed of a set of rules which act in sequence, and include models such as Bayesian rule lists (Letham et al., 2015) or the oneR algorithm (Holte, 1993). Rule trees are similar to rule lists, but allow branching after rules. This includes models such as CART decision trees (Breiman et al., 1984). Algebraic models take a final form of simple algebraic expressions, such as supersparse linear integer models (Ustun & Rudin, 2016).

Figure 1: Examples of different supported model forms. The bottom of each box shows predictions of the corresponding model as a function of $X_1$ and $X_2$.

Acknowledgements

The code here heavily derives from the wonderful work of previous projects. In particular, we build upon the following repos and users: sklearn-expertsys - by Tamas Madl and Benedict based on original code by Ben Letham. We also based many rule-based models on skope-rules by the skope-rules team (including Nicolas Goix, Florian Gardin, Jean-Matthieu Schertzer, Bibi Ndiaye, and Ronan Gautier). We also build upon the rulefit repository by Christoph Molnar.

References


