treeheatr: an R package for interpretable decision tree visualizations

This manuscript (permalink) was automatically generated from trang1618/treeheatr-manuscript@d85153e on April 30, 2020.

Authors

- **Trang T. Le**
  0000-0003-3737-6565 · trang1618 · trang1618
  Department of Biostatistics, Epidemiology and Informatics, Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA 19104

- **Jason H. Moore**
  0000-0002-5015-1099 · EpistasisLab · moorejh
  Department of Biostatistics, Epidemiology and Informatics, Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA 19104 · Funded by National Institutes of Health Grant Nos. [LM010098, LM012601, AI116794]

Abstract

Summary

treeheatr is an R package for creating interpretable decision tree visualizations with the data represented as heatmaps at the tree's terminal nodes. Going beyond the if-then step-by-step logic of a decision tree, the inclusion of a heatmap offers a broader view of the classification or regression problem and provides meaningful clarification of different node splits in the tree. Working harmoniously with other packages, *treeheatr* empowers the user with refined controls over the statistical threshold and presentation of the tree and heatmap.

Availability and implementation

The *treeheatr* package is freely available under the permissive MIT license at https://trang1618.github.io/treeheatr. It comes with a detailed vignette that is automatically built with continuous integration.

Contact

ttle@pennmedicine.upenn.edu

Supplementary information

Introduction

Tree-based algorithms such as random forests and gradient boosted trees are widely used techniques that comprise an important section of supervised machine learning. Visualizing and intepreting their building blocks, the single decision trees, are the first steps toward understanding these complex tree-based structures. However, it is difficult to incorporate the tree's predictive performance and the feature space in a single visualization. Existing softwares frequently treat all nodes in a decision tree similarly, leaving limited options for improving information presentation at the leaf nodes. Specifically, state-of-the-art libraries such as Python's dtreeviz, while producing aesthetic trees with detailed histograms at inner nodes, draw pie chart at leaf nodes. The *ggparty* R package allows the user to have full control of the representation of each node but fixes the terminal node widths, which can limit the ability to show more collective visualizations.

We have developed the *treeheatr* package to utilize the leaf node space to show the data as a heatmap where the samples and features are optionally clustered to improve interpretation. After simple installation, the user can apply *treeheatr* on their classification or regression problem with a single function, `heat_tree()`, as follows:

```
heat_tree(data, task = 'classification', target_lab = 'Outcome')
```

This one line of code above will generate the conditional inference tree, perform clustering, and produce a *ggplot* object that can be viewed in RStudio's viewer pane, saved to a graphic file, or embedded in an RMarkdown document. This example assumes a classification problem, but one can also apply *treeheatr* on a regression problem by changing the `task` argument.

This article is organized as follows. In Section 2, we describe the important functions and corresponding arguments in *treeheatr*. We demonstrate the flexibility the user has in tweaking these arguments to enhance understanding of the tree-based models applied on their dataset. In Section 3, we apply the *treeheatr* package to a real-world clinical dataset from a study of diabetes mellitus in a high risk population of Pima Indians [1]. [Finally, we discuss general guidelines for creating effective decision tree-heatmap visualization.]

Methods

We utilize the *ggparty* R package to compute the conditional inference tree (indirectly via the *partykit* R package) and edge and node information. However, because *ggparty* assumes fixed terminal widths, we recompute the node layout to accommodate the different number of samples shown in the heatmap at each terminal node. We implemented a node layout structure that can generalize as the trees grow in size. This new layout weighs the *x*-coordinate of the parent node according to the level of the child nodes in order to avoid crossing of tree branches. This relative weight can be adjusted with the lev_fac parameter in `heat_tree()`. `lev_fac = 1` sets the parent node's *x*-coordinate perfectly in the middle of those of its child nodes. The default `lev_fac = 1.3` seems to provide aesthetically pleasing trees independent of the tree size (see vignette).

As default, *treeheatr* automatically performs clustering when organizing the heatmap. To order the features, clustering is run on the two groups of features, continuous and categorical, across all samples (including the outcome label, unless `clust_target = FALSE`). To order the samples, clustering is run on samples within each terminal node of all features. *treeheatr* uses `cluster::daisy()` with the Gower metric [2] to incorporate both continuous and nominal categorical feature types.

In a visualization, it is difficult to find the balance between enhancing understanding and overloading information. We left it for the user to decide what type of information they want to show at inner nodes via different *geom* objects in the *ggparty* package. We believe displaying a heatmap at the terminal node space

This visualization nicely complements the current techniques of visualizing decision trees. Node purity, a metric measuring the tree's performance, can be visualized from the distribution of true outcome label at each terminal node in the first row. Comparing these values with the terminal node label gives a visual estimate of how accurate the tree predictions are. Without specifically choose two features to show in a 2-D scatter plot, we can infer correlation structures among features in the heatmap. The additional clustering also helps with the interpretation of the [...]

A simple example

This example visualizes the conditional inference tree model built to predict whether or not a patient has diabetes from a dataset provided by the National Institute of Diabetes and Digestive and Kidney Diseases [1]. This dataset of female patients at least 21 years old of Pima Indian heritage near Phoenix, Arizona was downloaded from Kaggle and has eight features: age, number of pregnancies, plasma glucose concentration, diastolic blood pressure, skin fold thickness, 2-hour serum insulin, body mass index and diabetes pedigree function. Detailed descriptions of these variables and data source can be found on the Kaggle page.

The following lines of code computes and visualizes the conditional decision tree along with the heatmap containing features that are important for building this model (Fig. 1):

```
heat_tree(
  data = diabetes,
  target_lab = 'Diabetes status',
  label_map = c(`0` = 'Negative', `1` = 'Positive')
)
```

The `heat_tree()` function takes a data frame, a character string indicating the column name associated with the outcome/phenotype (e.g., Diabetes status) and other optional arguments such as the mapping of the outcome label.

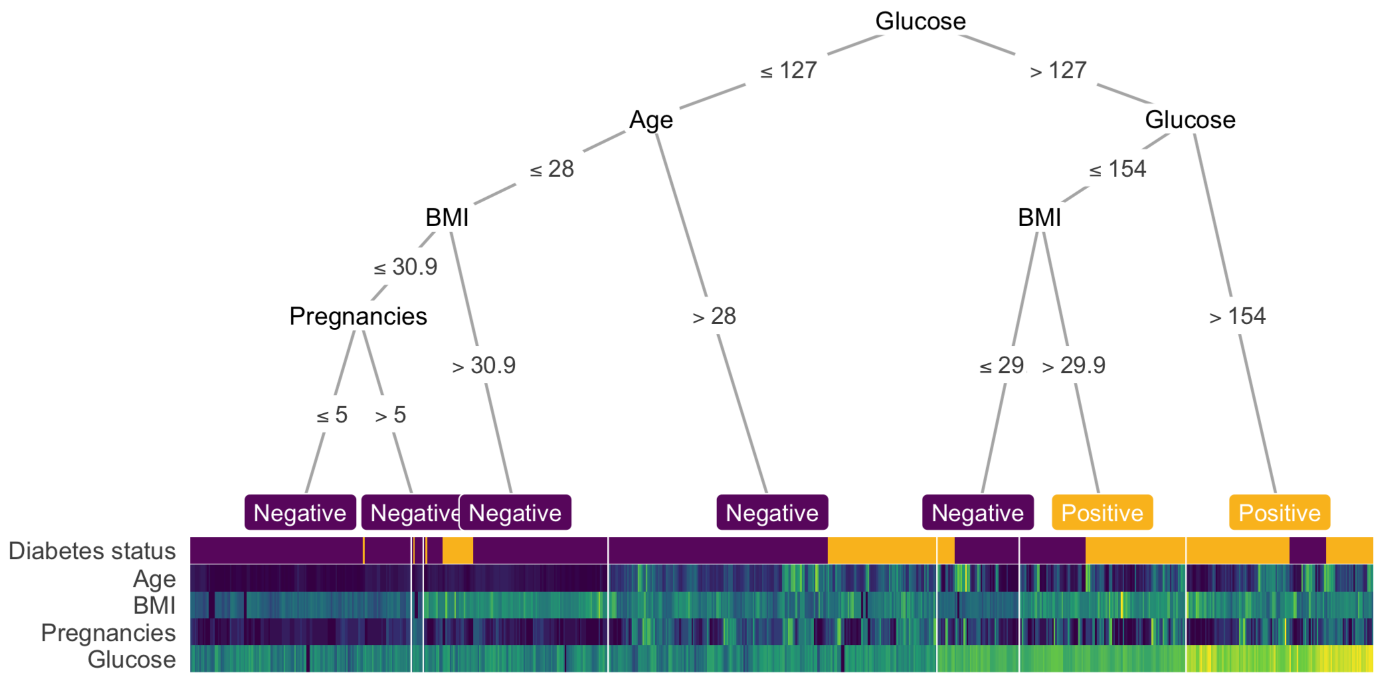

Figure 1: A decision tree-heatmap for predicting whether an individual has diabetes.

Conclusion

In this paper, we presented the new type of integrated visualization of decision trees and heatmaps, which provides a comprehensive data overview as well as model interpretation. We demonstrated that [...] The visualization is implemented in an easily installed package with a detailed vignette. Released as open source software, {treeheatr} hopes to receive contribution from other developers.

Future works on *treeheatr* include enhancements such as supporting heatmap visualization of a holdout set and highlighting the tree branches that point to a specific holdout sample.

Acknowledgements

The *treeheatr* package was made possible by leveraging integral R packages including *ggplot2* [3], *partykit* [4], *ggparty* [5] and many others. We would also like to thank Daniel Himmelstein for his helpful comments on the package's continuous integration configuration.

References

1. **Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus**
   Jack W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, R. S. Johannes
   Proceedings of the Annual Symposium on Computer Application in Medical Care (1988-11-09)
   https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/
   PMCID: PMC2245318

2. **A General Coefficient of Similarity and Some of Its Properties**
   J. C. Gower
   Biometrics (1971-12) https://doi.org/dxtrt7
   DOI: 10.2307/2528823

3. **Ggplot2: elegant graphics for data analysis**
   Hadley Wickham
   Springer (2009)
   ISBN: 9780387981406

4. **partykit: A Modular Toolkit for Recursive Partytioning in R**
   Torsten Hothorn, Achim Zeileis
   Journal of Machine Learning Research (2015) http://jmlr.org/papers/v16/hothorn15a.html

5. **martin-borkovec/ggparty**
   Martin Borkovec
   (2020-04-25) https://github.com/martin-borkovec/ggparty