Master Thesis - Explainable Machine Learning - Visualization of Random Forests

Fabio Rougier

October 11, 2022

1 Summary

DO SUMMARY AT THE END

Contents

2 Introduction

Random Forests (RF) are a powerful ensemble method with a low barrier of entry. Because of their ease of use and performance they are used in many applications. However, they fall short when it comes to transparency. On the one hand RFs can offer a multitude of valuable insights into their decision making and the data they process. On the other hand visualizing this is usually a challenge because of the inherent scale of a RF and the aggregation of their decision making process. The goal of this work was to provide relative beginners with a tool to explore a RF on a detailed level.

2.1 Visualizing Decision Trees

An obvious approach to visualizing a RF is inspecting the decision trees that constitute the RF. One example for the visualization of a decision tree is *BaobabView* (Van Den Elzen & Van Wijk, 2011). As many other approaches visualizing decision trees, it utilizes Node-Link Diagrams (NLDs) as its' main visualization. A confusion matrix yields additional insights into the relations of the underlying features. However this visualization does fall short when trees grow too large, as it becomes hard to inspect every individual node and branch of the tree. This is one of the problems that *TaxonTree* tries to overcome (Parr, Lee, Campbell, & Bederson, 2003). It uses a tree visualization approach that is scalable for large trees by adding the possibility to zoom, browse and search the tree. While this does help if a tree grows too large, it does not provide any insight on the general structure of the tree as a whole. Generalizing either of the approaches towards RFs is also non trivial. Both simply lack the scalability in the desired dimension. It also leads into a dangerous territory of focusing too much on the structure of individual trees. RFs - as all ensemble methods - arrive at their decisions by combining the decisions of the individual trees. Inspecting and even understanding individual trees, will only yield limited insights over the RF.

2.2 Visualizing Random Forests

To fully understand the structure and decision making of a RF, many aspects of the RF have to be conveyed by the visualization. First of all typical indicators like a *F1 score* give an indication of the overall performance. Additionally there are some metrics specific to RFs that should be included to get a deeper understanding of a particular instance of the RF, like the *mean impurity* of the individual trees or the *out of bag error*. With its' unique way of providing a distance measure for features, a confusion matrix can

also yield valuable insights to the relations of the features and how the RF interprets them. As stated before, the visualization of RFs heavily relies on how it overcomes the scalability issues of RFs.

3 Related Work

3.1 ReFine

This approach is rather old, but still worth mentioning, because it highlights the fact, that visualizing RFs has been a challenge for some time now (Kuznetsova, Westenberg, Buchin, Dinkla, & van den Elzen, 2014). In this particular case, the technical implementation is of course out-dated, but the ideas are still relevant. *ReFine* uses a small multiples view of the trees to give the user insights from different angles. The main visualization uses icicle plots, as the author deems them most suitable due to their efficient use of space. While this does allow to show a considerable number of trees in one view, it does not scale well enough for large amounts of trees in a RF. An important distinction raised by the author is that there are different users for RFs, with very different requirements towards a RF visualization. While a machine learning expert might be more focused on improving model performance, an analyst or domain expert would be more concerned about the insights provided by the models.

3.2 iForest

The *iForest* visualization is one of the most promising visualization approaches for RFs. (Zhao, Wu, Lee, & Cui, 2018) It focuses on the interpretability of the RF and raises the concern that many domains would not even consider using RFs because of their lack thereof. The authors also approach the understanding of RFs by coming from two different angles: *Feature Analysis* and *Case Based Reasoning*. Both require different charts and give the user valuable insight into the dataset and the RF. The elaborate, dashboard-like web application utilizes the benefits of small multiples and has interconnected and interactive charts, each devoted to offer a specific perspective. One especially unique part of the dashboard is the use of *Partial Dependence Plots* to display the RF's classification behavior in regard to each feature. This is supported by a bar chart of the feature's distribution in order to support this view with more context. While the dashboard is a powerful tool, it can be overwhelming for users. It also requires some in-depth knowledge about RFs in order to come to conclusions or even an

actionable instruction. *iForest* is clearly aimed at supporting data scientists to understand their own creations and not suited for a domain expert.

3.3 ExMatrix

The *ExMatrix* follows an out-of-the-box-thinking approach by breaking up machine learning models into a set of rules (Neto & Paulovich, 2020) (Ming, Qu, & Bertini, 2018). It is therefore even more flexible because it is not limited to be used with RFs only, but could be applied to almost any kind of machine learning model. It is best suited to abstract ensemble models and simplify them. This is however not to be mistaken with creating surrogates from the given models, as it reflects the underlying models exactly. By breaking down a RF in said rules, it is possible to display the entire RF in a matrix structure. It does this by representing columns as features and rows as rules, resulting in cells as so-called *rules predicates*. This entire rethinking of the RF as a whole allows for unique graphs and insights in the RFs' decision making. Thinking of the RF as a set of comparable rules and evaluating those, yields a very deep understanding, both on a case-based sample level, but also on a global level. While not supported in the original paper, this would also allow for intricate comparisons of models on the same data set. The biggest issue with this approach is, that while it does yield powerful insights, it can be very difficult to convey these insights to a domain expert. While a data scientist can be expected to wrap their head around the idea of disassembling a RF into a set of rules, this idea is not intuitive for a domain expert who might already have trouble understanding how the RF works in the first place.

3.4 Summary

While the main challenge of gaining access to the insights and details of the inner workings of a RF might have been solved already they seem to be hidden behind a kind of complexity layer. Powerful approaches for RF visualization exist, but there is a need to make them more accessible for domain experts. Considering not only the audience, but also the intended use of the visualization is paramount for the visualization to be useful. There is a lot of value to be derived from the existing work and some of the ideas have influenced this work.

4 Methodology

4.1 Python

The implementation of this work was done in *Python 3.10.4* (Van Rossum & Drake, 2009), as it has become one of the standard programming languages for data science and machine learning. This allowed the usage of many commonly used libraries, like *NumPy* (Harris et al., 2020), *pandas* (Wes McKinney, 2010), *scikit-Learn* (Pedregosa et al., 2011) and more. Using Python also opens up the possibility to further extend this work in the future, while the libraries and possibilities of Python keep improving and expanding. Some of the libraries used in this work make use of *Cython* (Behnel et al., 2011) to speed up the execution of certain functions. This was particularly relevant for this work, as will be elaborated in a later section. We will go in depth on some of the libraries used in the following sections.

4.2 pandas

The *pandas* library offers a powerful data structure called *DataFrame*, which in essence is just a table. Its' implementation is made so convenient and efficient though, that it allows for both intuitive and performant data manipulation.
In this work, *pandas* is used as a backbone throughout the entire lifecycle of the visualization.

4.3 scikit-Learn

With the *scikit-Learn* library, machine learning is made easily accessible, even to beginners, while still providing some more advanced configurations if necessary.
For this work, both the *RandomForestClassifier* and the *DecisionTreeClassifier* are a main part of the processes in the background.

4.4 Streamlit

Streamlit provides users with an incredibly easy entry point to transferring Python visualizations into the web browser. With a simple *"st.write()"* command, any text, visual or even DataFrame can be displayed in the browser. While it certainly has its' limits, it is a great tool for a fast development of dashboards and visualizations.

Streamlit was used in this work to handle the web application and the user interface that enables interactions with the visualizations and algorithms.

4.5 Vega-Altair

Speaking of visualizations, *Vega-Altair* (VanderPlas et al., 2018) is a great library for creating interactive charts, with a great deal of flexibility for customization. It is built on top of *Vega-Lite* (Satyanarayan, Moritz, Wongsuphasawat, & Heer, 2017), which is a declarative grammar using the JSON standard for building graphs, that are then rendered using the *Vega-Lite* compiler. *Vega-Altair*, or *Altair* for short, is especially unique in its' way of constructing graph, by using *encodings* for each visual element. This syntax is very intuitive to use makes the library both powerful and easy to use.
Every chart in this work is created using *Altair*, which in turn allows the use of *Altair's* interactive capabilities like brushing and linking.

4.6 NetworkX

NetworkX (Hagberg, Schult, & Swart, 2008) is one of the most popular libraries for manipulating graphs in Python. It contains many useful functions for graph manipulation.
Decision Trees are essentially binary trees, which can be represented as graphs. As such, *NetworkX* was used for some particular graph operations like the graph edit distance. In order to transform the *DecisionTreeClassifier* into a graph, *PyGraphviz* was used as an intermediate step.

References

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science & Engineering*, *13*(2), 31–39.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference* (p. 11 - 15). Pasadena, CA USA.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020, September). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Retrieved from `https://doi.org/10.1038/s41586-020-2649-2` doi: 10.1038/s41586-020-2649-2

Kuznetsova, N., Westenberg, M., Buchin, K., Dinkla, K., & van den Elzen, S. (2014). Random forest visualization. *Eindhoven University of Technology*.

Ming, Y., Qu, H., & Bertini, E. (2018). Rulematrix: Visualizing and understanding classifiers with rules. *IEEE transactions on visualization and computer graphics*, *25*(1), 342–352.

Neto, M. P., & Paulovich, F. V. (2020). Explainable matrix-visualization for global and local interpretability of random forest classification ensembles. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 1427–1437.

Parr, C. S., Lee, B., Campbell, D., & Bederson, B. B. (2003). *Taxontree: Visualizing biodiversity information* (Tech. Rep.). MARYLAND UNIV COLLEGE PARK INST FOR ADVANCED COMPUTER STUDIES.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., & Heer, J. (2017). Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, *23*(1), 341-350.

Van Den Elzen, S., & Van Wijk, J. J. (2011). Baobabview: Interactive construction and analysis of decision trees. In *2011 ieee conference on visual analytics science and technology (vast)* (pp. 151–160).

VanderPlas, J., Granger, B., Heer, J., Moritz, D., Wongsuphasawat, K., Satyanarayan, A., ... Sievert, S. (2018). Altair: Interactive statistical visualizations for python. *Journal of Open Source Software*, *3*(32), 1057. Retrieved from `https://doi.org/10.21105/joss.01057` doi: 10.21105/joss.01057

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Wes McKinney. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (p. 56 - 61). doi: 10.25080/Majora-92bf1922-00a

Zhao, X., Wu, Y., Lee, D. L., & Cui, W. (2018). iforest: Interpreting random forests via visual analytics. *IEEE transactions on visualization and computer graphics*, *25*(1), 407–416.