SIRUS.jl: Interpretable Machine Learning via Rule Extraction

Rik Huijzer [1¶], **Frank Blaauw** [2], **Ruud J. R. den Hartigh** [1], and **Peter de Jonge** [1]

1 University of Groningen, Groningen, the Netherlands **2** Researchable, Assen, the Netherlands ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

DRAFT

Summary

`SIRUS.jl` is a pure Julia implementation of the original Stable and Interpretable RUle Sets (SIRUS) algorithm. The SIRUS algorithm is a fully interpretable version of random forests, that is, it reduces the large amount of trees in the forest to a small amount of interpretable rules. With our Julia implementation, we aimed to reproduce the original C++ and R implementation in a high-level language to verify the algorithm as well as making the code easier to read. Furthermore, we made the code available under the permissive MIT license. In turn, this allows others to research the algorithm further or easily port it to production environments.

Statement of need

Due to the succesful applications of neural networks in various domains, there has been an paradigm shift within the field of machine learning towards the use of non-interpretable models, also known as "black box" models. This is appropriate for low stakes domains such as spam detection and product recommendations, but can be problematic in high stakes domains where model decisions have a real-world impact on individuals. In such situations, black box models may lead to unsafe or unreliable predictions (Barredo Arrieta et al., 2020; Doshi-Velez & Kim, 2017). However, the set of fully interpretable models is often limited to linear models and decision trees. Linear models tend to perform poorly when the data does not satisfy suitable distributions and decision trees perform poorly compared to random forests. Random forests (Breiman, 2001), often outperform linear models and random forests, but are not fully interpretable. Visualization techniques, such as SHAP (Lundberg & Lee, 2017), allow inspection of feature importances, but do not provide enough information to reproduce the predictions made by the model. The SIRUS algorithm solves these issues by first restricting the split points in the random forest algorithm to a stable subset of points, and by then extracting a small and interpretable rule set (Bénard et al., 2021). However, the original SIRUS algorithm was implemented in C++ and R, which makes it hard to inspect and extend. An implementation in one high-level language allows verification of the algorithm and allows researchers to investigate further algorithmic improvements. Furthermore, the original algorithm was covered by a copyleft license meaning that copies are required to be made freely available. A more permissive license makes it easier to port the algorithm to other languages or move it to production environments.

Interpretability

To show that the algorithm is fully interpretable, we fitted the model on Haberman's Survival Dataset (Haberman, 1999). The dataset contains survival data on patients who had undergone surgery for breast cancer and contains three features, namely the number of auxillary nodes that were detected, the age of the patient at the time of the operation, and the patient's year of operation.

```
StableRules model with 8 rules:
 if X[i, :nodes] < 8.0 then 0.156 else 0.031 +
 if X[i, :nodes] < 14.0 then 0.164 else 0.026 +
 if X[i, :nodes] < 4.0 then 0.128 else 0.037 +
 if X[i, :nodes] ≥ 8.0 & X[i, :age] < 38.0 then 0.0 else 0.008 +
 if X[i, :year] ≥ 1966.0 & X[i, :age] < 42.0 then 0.0 else 0.005 +
 if X[i, :nodes] < 2.0 then 0.107 else 0.034 +
 if X[i, :year] ≥ 1966.0 & X[i, :age] < 38.0 then 0.0 else 0.001 +
 if X[i, :year] < 1959.0 & X[i, :nodes] ≥ 2.0 then 0.0 else 0.003
and 2 classes: [0.0, 1.0].
Note: showing only the probability for class 1.0 since class 0.0 has
      probability 1 - p.
```

This shows that the model contains 8 rules. The first rule, for example, can be interpreted as: *If the number of detected auxillary nodes is lower than 8, then take 0.156, otherwise take 0.031.*

This is done for all 8 rules and the total score is summed to get a prediction. In essence, the first rule says that if there are less than 8 auxillary nodes detected, then the patient will most likely survive (`class == 1.0`). In essence, the model states that if there are many auxillary nodes detected, then it is (unfortunately) less likely that the patient will survive.

This model is fully interpretable because there are few rules which can all be interpreted in isolation reasonably well. Random forests, in contrasts, consist of hundreds to thousands of trees, which are not interpretable due to the large amount of trees. A common workaround for this is to use SHAP or Shapley values to visualize the fitted model. The problem with those methods is that they do not allow full reproducibility of the predictions. For example, if we would inspect the fitted model on the aforementioned Haberman dataset via SHAP, then we could learn feature importances. In practice that would mean that we could tell which features were important. In many real-world situations this is not enough. Imagine having to tell a patient that was misdiagnosed by the model: "Sorry about our prediction, we were wrong and we didn't really know why. Only that nodes is an important feature in the model, but we don't know whether this played a large role in your situation."

Stability

Another problem that the SIRUS algorithm solves is that of model stability. A stable model is defined as a model which leads to similar conclusions for small changes to data (Yu, 2020). Unstable models can be difficult to apply in practice since they might require processes to constantly change. Also, they are considered less trustworthy.

Having said that, most statistical models are quite stable since a higher stability is often correlated to a higher predictive performance. Put differently, an unstable model by definition leads to different conclusions for small changes to the data and, hence, small changes to the data can cause a sudden drop in predictive performance. One model which suffers from a low stability is a decision tree. This is because a decision tree will first create the root node of the tree, so a small change in the data can cause the root, and therefore the rest, of the tree to be completely different. The SIRUS algorithm has solved the instability of random forests by "stabilizing the trees" (Bénard et al., 2021) and the authors have proven mathematically that the stabilization works.

DRAFT

Predictive Performance

The model is based on random forests and therefore has excellent performance in settings where the number of variables is comparatively large to the number of datapoints (Biau and Scornet). The algorithm converts a large number of trees to a small number of rules to improve interpretability. This tradeoff comes at a small performance cost. For example, the cross-validated scores on the Haberman dataset are listed in Table 1.

Model	AUC ± 1.96SE	Interpret-ability
`LGBMClassifier()`	0.71 ± 0.06	Medium
`LGBMClassifier(; max_depth=2)`	0.67 ± 0.06	Medium
`DecisionTreeClassifier(; max_depth=2)`	0.63 ± 0.06	High
`StableRulesClassifier(; max_depth=2)`	$\mathbf{0.71 \pm 0.05}$	**High**
`StableRulesClassifier(; max_depth=2, max_rules=25)`	$\mathbf{0.70 \pm 0.09}$	**High**
`StableRulesClassifier(; max_depth=2, max_rules=10)`	$\mathbf{0.67 \pm 0.07}$	**High**
`StableRulesClassifier(; max_depth=1, max_rules=25)`	$\mathbf{0.67 \pm 0.07}$	**High**

Table 1: Predictive performance in terms of Area Under the Curve (AUC) score for the LightGBM, decision tree, and SIRUS models.

This shows that the SIRUS algorithm performs very comparable to the state-of-the-art LGBM classifier by Microsoft. The tree depths are set to at most 2 because rules which belong to a depth of 3 will (almost) never show up in the final model.

Code Example

The model can be used via the `MLJ.jl` (Blaom et al., 2020) machine learning interface. For example, this is the code used to fit the model on the full Haberman dataset:

```
model = StableRulesClassifier(; max_depth=2, max_rules=8)
mach = machine(model, X, y)
fit!(mach)
```

and model performance was estimated via cross-validation (`CV`):

```
resampling = CV(; nfolds=10, shuffle=true)
evaluate(model, X, y; resampling, measure=auc)
```

Funding

This research was supported by the Ministry of Defence, the Netherlands.

Acknowledgements

We thank Clément Bénard for his help in re-implementing the SIRUS algorithm. Furthermore, we thank Anthony Bloam and Dávid Hanák (Cursor Insight) for respectively doing code reviews and fixing a critical bug.

References

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., & others. (2020). Explainable Artificial

DRAFT

Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Bénard, C., Biau, G., Veiga, S. D., & Scornet, E. (2021). SIRUS: Stable and Interpretable RUle Set for classification. *Electronic Journal of Statistics*, *15*(1), 427–505. https://doi.org/10.1214/20-EJS1792

Blaom, A. D., Kiraly, F., Lienart, T., Simillides, Y., Arenas, D., & Vollmer, S. J. (2020). MLJ: A julia package for composable machine learning. *Journal of Open Source Software*, *5*(55), 2704. https://doi.org/10.21105/joss.02704

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1023/A:1010933404324

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv Preprint arXiv:1702.08608*. https://doi.org/10.48550/arXiv.1702.08608

Haberman, S. (1999). *Habermanś Survival*. UCI Machine Learning Repository. https://doi.org/10.24432/C5XK51

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

DRAFT