The Journal of Open Source Software

SIRUS.jl: Interpretable Machine Learning via Rule Extraction

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

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

DOI: 10.21105/joss.05786

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @sylvaticus
- @gdalle

Submitted: 07 July 2023
Published: 11 October 2023

Summary

SIRUS.jl[1] is an implementation of the original Stable and Interpretable RUle Sets (SIRUS) algorithm in the Julia programming language (Bezanson et al., 2017). The SIRUS algorithm is a fully interpretable version of random forests, that is, it reduces thousands of trees in the forest to a much lower number of interpretable rules (e.g., 10 or 20). 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. We show that the model performs well on classification tasks while retaining interpretability and stability. 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 systems.

Statement of need

Many of the modern day machine learning models are noninterpretable models, also known as *black box* models. Well-known examples of noninterpretable models are random forests (Breiman, 2001) and neural networks. Such models are available in the Julia programming language via, for example, LightGBM.jl (Ke et al., 2017), Flux.jl (Innes, 2018), and BetaML.jl (Lobianco, 2021). Although these models can obtain high predictive performance and are commonly used, they can be problematic in high stakes domains where model decisions have real-world impact on individuals, such as suggesting treatments or selecting personnel. The reason is that noninterpretable models may lead to unsafe, unfair, or unreliable predictions (Barredo Arrieta et al., 2020; Doshi-Velez & Kim, 2017). Furthermore, interpretable models may allow researchers to learn more from the model, which in turn may allow researchers to make better model decisions and achieve a higher predictive performance.

However, the set of interpretable models is often limited to ordinary and generalized regression models, decision trees, RuleFit, naive Bayes classification, and k-nearest neighbors (Molnar, 2022). For these models, however, predictive performance can be poor for certain tasks. Linear models, for instance, may perform poorly when features are correlated and can be sensitive to the choice of hyperparameters. For decision trees, predictive performance is poor compared to random forests (James et al., 2013). RuleFit is not available in Julia and is *unstable* (Bénard, Biau, Veiga, et al., 2021), meaning sensitive to small changes in data. Naive Bayes, available in Julia as NaiveBayes.jl[2], is often overlooked and can be a suitable solution, but only if the features are independent (Ashari et al., 2013).

Researchers have attempted to make the random forest models more interpretable. Model interpretation techniques, such as SHAP (Lundberg & Lee, 2017) or Shapley, available via

[1] Source code available at https://github.com/rikhuijzer/SIRUS.jl.
[2] Source code available at https://github.com/dfdx/NaiveBayes.jl.

Shapley.jl[3], have been used to visualize the fitted model. However, the disadvantage of these techniques are that they convert the complex model to a simplified representation. This causes the simplified representation to be different from the complex model and may therefore hide biases and issues related to safety and reliability (Barredo Arrieta et al., 2020).

The SIRUS algorithm solves this by simplifying the complex model and by then using the simplified model for predictions. This ensures that the same model is used for interpretation and prediction. However, the original SIRUS algorithm was implemented in about 10k lines of C++ and 2k lines of R code[4] which makes it hard to inspect and extend due to the combination of two languages. Our implementation is written in about 2k lines of pure Julia code. This allows researchers to more easily verify the algorithm and investigate further improvements. Furthermore, the original algorithm was covered by the GPL-3 copyleft license meaning that copies are required to be made freely available. A more permissive license makes it easier to port the code to other languages or production systems.

Interpretability

To show that the algorithm is fully interpretable, we fit an example on the 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 axillary *nodes* that were detected, the *age* of the patient at the time of the operation, and the patient's *year* of operation. For this example, we have set the hyperparameters for the maximum number of rules to 8 since this is a reasonable trade-off between predictive performance and interpretability. Generally, a higher maximum number of rules will yield a higher predictive performance. We have also set the maximum depth hyperparameter to 2. This hyperparameter means that the random forests inside the algorithm are not allowed to have a depth higher than 2. In turn, this means that rules contain at most 2 clauses (`if A & B`). When the maximum depth is set to 1, then the rules contain at most 1 clause (`if A`). Most rule-based models, including SIRUS, are restricted to depth of 1 or 2 (Bénard, Biau, Veiga, et al., 2021).

The output for the fitted model looks as follows (see Section *Code Example* for the code):

```
StableRules model with 8 rules:
 if X[i, :nodes] < 7.0 then 0.238 else 0.046 +
 if X[i, :nodes] < 2.0 then 0.183 else 0.055 +
 if X[i, :age] ≥ 62.0 & X[i, :year] < 1959.0 then 0.0 else 0.001 +
 if X[i, :year] < 1959.0 & X[i, :nodes] ≥ 2.0 then 0.0 else 0.006 +
 if X[i, :nodes] ≥ 7.0 & X[i, :age] ≥ 62.0 then 0.0 else 0.008 +
 if X[i, :year] < 1959.0 & X[i, :nodes] ≥ 7.0 then 0.0 else 0.003 +
 if X[i, :year] ≥ 1966.0 & X[i, :age] < 42.0 then 0.0 else 0.008 +
 if X[i, :nodes] ≥ 7.0 & X[i, :age] ≥ 42.0 then 0.014 else 0.045
and 2 classes: [0, 1].
```

This shows that the model contains 8 rules where the first rule, for example, can be interpreted as:

If the number of detected axillary nodes is lower than 7, then take 0.238, otherwise take 0.046.

This calculation is done for all 8 rules and the score is summed to get a prediction. In essence, the first rule says that if there are less than 8 axillary nodes detected, then the patient is more likely to survive (`class == 1`). Put differently, the model states that if there are many axillary nodes detected, then it is, unfortunately, less likely that the patient will survive. This model is fully interpretable because the model contains a few dozen rules which can all be interpreted in isolation and together.

[3]Source code available at https://gitlab.com/ExpandingMan/Shapley.jl.
[4]Source code available at https://gitlab.com/drti/sirus.

Stability

Another problem that the SIRUS algorithm addresses 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 as they might require processes to constantly change. This also makes such models appear less trustworthy. Put differently, an unstable model by definition leads to different conclusions for small changes to the data and, hence, small changes to the data could cause a sudden drop in predictive performance. One model which suffers from a low stability is a decision tree, available via DecisionTree.jl (Sadeghi et al., 2022), because it 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 (Molnar, 2022). Similarly, linear models can be highly sensitive to correlated data and, in the case of regularized linear models, the choice of hyperparameters. The aforementioned RuleFit algorithm also suffers from stability issues due to the unstable combination of tree fitting and rule extraction (Bénard, Biau, Veiga, et al., 2021). The SIRUS algorithm solves this problem by stabilizing the trees inside the forest, and the original authors have proven the correctness of this stabilization mathematically (Bénard, Biau, Veiga, et al., 2021). In the rest of this paper, we will compare the predictive performance of SIRUS.jl to the performance of decision trees (Sadeghi et al., 2022), linear models, XGBoost (Chen & Guestrin, 2016), and the original (C++/R) SIRUS implementation (Bénard, Biau, Veiga, et al., 2021). The interpretability and stability are summarized in Table 1.

	Decision Tree	**Linear Model**	**XGBoost**	**SIRUS**
Interpretability	High	High	Medium	High
Stability	Low	Medium	High	High

Table 1: Summary of interpretability and stability for various models.

Predictive Performance

The SIRUS model is based on random forests and therefore well suited for settings where the number of variables is comparatively large to the number of datapoints (Biau & Scornet, 2016). To make the random forests interpretable, the large number of trees are converted to a small number of rules. The conversion works by converting each tree to a set of rules and then pruning the rules by removing simple duplicates and linearly dependent duplicates, see the SIRUS.jl documentation or the original paper (Bénard, Biau, Da Veiga, et al., 2021) for details. In practice, this trade-off between between model complexity and interpretability comes at a small performance cost.

To show the performance, we compared SIRUS to a decision tree, linear model, XGBoost, and the original (C++/R) SIRUS algorithm; similar to Table 1. We have used Julia version 1.9.3 with SIRUS version 1.3.3 (at commit `5c87eda`), 10-fold cross-validation, and we will present variability as $1.96 * \text{standard error}$ for all evaluations with respectively the following datasets, outcome variable type, and measures: Haberman's Survival Dataset (Haberman, 1999) binary classification dataset with AUC, Titanic (Eaton & Haas, 1995) binary classification dataset with Area Under the Curve (AUC), Breast Cancer Wisconsin (Wolberg & Street, 1995) binary classification dataset with AUC, Pima Indians Diabetes (Smith et al., 1988) binary classification dataset with AUC, Iris (Fisher, 1936) multiclass classification dataset with accuracy, and Boston Housing (Harrison & Rubinfeld, 1978) regression dataset with R^2; see Table 2. For full details, see `test/mlj.jl`. The performance scores were taken from the SIRUS.jl test job that ran following commit `5c873da` using GitHub Actions. The result for the Iris dataset for the original SIRUS algorithm is missing because the original algorithm has not implemented multiclass classification.

At the time of writing, SIRUS's predictive performance is comparable to the linear model and

Dataset	Decision Tree	Linear Model	XGBoost max depth: ∞	XGBoost max depth: 2	Original SIRUS max depth: 2 max rules: 10	SIRUS.jl max depth: 2 max rules: 10
Haberman	0.54 ± 0.06	0.69 ± 0.06	0.65 ± 0.04	0.63 ± 0.04	0.66 ± 0.05	0.67 ± 0.06
Titanic	0.76 ± 0.05	0.84 ± 0.02	0.86 ± 0.03	0.87 ± 0.03	0.81 ± 0.02	0.83 ± 0.02
Cancer	0.92 ± 0.03	0.98 ± 0.01	0.99 ± 0.00	0.99 ± 0.00	0.96 ± 0.02	0.98 ± 0.01
Diabetes	0.67 ± 0.05	0.70 ± 0.06	0.80 ± 0.04	0.82 ± 0.03	0.80 ± 0.02	0.75 ± 0.05
Iris	0.95 ± 0.03	0.97 ± 0.03	0.94 ± 0.04	0.93 ± 0.04		0.77 ± 0.08
Boston	0.74 ± 0.11	0.70 ± 0.05	0.87 ± 0.05	0.86 ± 0.05	0.63 ± 0.07	0.61 ± 0.09

Table 2: Predictive performance estimates.

XGBoost on the binary classification datasets, that is, Haberman, Titanic, Breast Cancer, and Diabetes. The best performance occurs at the Diabetes dataset where both XGBoost and the SIRUS models outperform the linear model. The reason for this could be that negative effects are often nonlinear for fragile systems (Taleb, 2020). For example, it could be that an increase in oral glucose tolerance increases the chance of diabetes exponentially. In such cases, the hard cutoff points chosen by tree-based models, such as XGBoost and SIRUS, may fit the data better.

For the multiclass Iris classification and the Boston Housing regression datasets, the performance was worse than the other non-SIRUS models. It could be that this is caused by a bug in the implementation or because this is a fundamental issue in the algorithm. Further work is needed to find the root cause or workarounds for these low scores. One possible solution would be to add SymbolicRegression.jl (Cranmer, 2023) as a secondary back end for regression tasks. Similar to SIRUS.jl, SymbolicRegression.jl can fit expressions of a pre-defined form to data albeit with more free parameters, which might fit better but also might cause overfitting, depending on the data. This achieves performance that is similar to XGBoost (Hanson, 2023).

In conclusion, interpretability and stability are often required in high-stakes decision making contexts such as personnel or treatment selection. In such contexts and when the task is classification, SIRUS.jl obtains a reasonable predictive performance, while retaining model stability and interpretability.

Code Example

The model can be used via the Machine Learning Julia (MLJ) (Blaom et al., 2020) interface. The following code, for example, was used to obtain the fitted model for the Haberman example at the start of this paper, and is also available in the SIRUS.jl docs[5].

We first load the dependencies:

```
using CategoricalArrays: categorical
using CSV: CSV
using DataDeps: DataDeps, DataDep, @datadep_str
using DataFrames
using MLJ
using StableRNGs: StableRNG
using SIRUS: StableRulesClassifier
```

And specify the Haberman dataset via DataDeps.jl, which allows data verification via the checksum and enables caching:

[5] https://sirus.jl.huijzer.xyz/dev/basic-example/

```
function register_haberman()
    name = "Haberman"
    message = "Haberman's Survival Data Set"
    remote_path = "https://github.com/rikhuijzer/haberman-survival-dataset/
        releases/download/v1.0.0/haberman.csv"
    checksum = "a7e9aeb249e11ac17c2b8ea4fdafd5c9392219d27cb819ffaeb8a869eb727a0f"
    DataDeps.register(DataDep(name, message, remote_path, checksum))
end
```

Next, we load the data into a `DataFrame`:

```
function load_haberman()::DataFrame
    register_haberman()
    path = joinpath(datadep"Haberman", "haberman.csv")
    df = CSV.read(path, DataFrame)
    df[!, :survival] = categorical(df.survival)
    return df
end
```

We split the data into features (`X`) and outcomes (`y`):

```
data = load_haberman()
X = select(data, Not(:survival))
y = data.survival
```

We define the model that we want to use with some reasonable hyperparameters for this small dataset:

```
model = StableRulesClassifier(; rng=StableRNG(1), q=4, max_depth=2, max_rules=8)
```

Finally, we fit the model to the data via MLJ and show the fitted model:

```
mach = let
    mach = machine(model, X, y)
    MLJ.fit!(mach)
end

mach.fitresult
```

Resulting in the fitresult that was presented at the start of this paper.

Funding

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

Acknowledgments

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

References

Ashari, A., Paryudi, I., & Tjoa, A. M. (2013). Performance comparison between naïve Bayes, decision tree and k-nearest neighbor in searching alternative design in an energy simulation tool. *International Journal of Advanced Computer Science and Applications*, *4*(11). https://doi.org/10.14569/IJACSA.2013.041105

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 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., Da Veiga, S., & Scornet, E. (2021). Interpretable random forests via rule extraction. *International Conference on Artificial Intelligence and Statistics*, 937–945.

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

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*, 197–227. https://doi.org/10.1007/s11749-016-0481-7

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

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cranmer, M. (2023). Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv Preprint arXiv:2305.01582*. https://doi.org/10.48550/arXiv.2305.01582

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

Eaton, J. P., & Haas, C. (1995). *Titanic: Triumph and tragedy*. WW Norton & Company.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*(2), 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

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

Hanson, E. (2023). *[ANN] SIRUS.jl v1.2: Interpretable machine learning via rule extraction*. https://discourse.julialang.org/t/ann-sirus-jl-v1-2-interpretable-machine-learning-via-rule-extraction/100932/3

Harrison, D., & Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, *5*(1), 81–102. https://doi.org/10.1016/0095-0696(78)90006-2

Innes, M. (2018). Flux: Elegant machine learning with Julia. *Journal of Open Source Software*, *3*(25), 602. https://doi.org/10.21105/joss.00602

James, G., Witten, D., Hastie, T., Tibshirani, R., & others. (2013). *An introduction to statistical learning* (Vol. 112). Springer. https://doi.org/10.1007/978-1-0716-1418-1

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*.

Lobianco, A. (2021). BetaML: The beta machine learning toolkit, a self-contained repository of machine learning algorithms in Julia. *Journal of Open Source Software*, *6*(60), 2849. https://doi.org/10.21105/joss.02849

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

Molnar, C. (2022). *Interpretable machine learning*.

Sadeghi, B., Chiarawongse, P., Squire, K., Jones, D. C., Noack, A., St-Jean, C., Huijzer, R., Schätzle, R., Butterworth, I., Peng, Y.-F., & Blaom, A. (2022). *DecisionTree.jl - a Julia implementation of the CART Decision Tree and Random Forest algorithms* (Version 0.12.3). Zenodo. https://doi.org/10.5281/zenodo.7359268

Smith, J. W., Everhart, J. E., Dickson, W., Knowler, W. C., & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, 261.

Taleb, N. N. (2020). Statistical consequences of fat tails: Real world preasymptotics, epistemology, and applications. *arXiv Preprint arXiv:2001.10488*. https://doi.org/10.48550/arXiv.2001.10488

Wolberg, M., William, & Street, W. (1995). *Breast Cancer Wisconsin (Diagnostic)*. UCI Machine Learning Repository. https://doi.org/10.24432/C5DW2B

Yu, B. (2020). Veridical data science. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 4–5. https://doi.org/10.1073/pnas.1901326117