UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] Hierarchical Shrinkage: Improving the Accuracy and Interpretability of Tree-Based Methods

Anonymous[1, ID]

[1] Anonymous Institution

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
03 February 2023

Published
–

DOI
–

Reproducibility Summary

Scope of Reproducibility — The paper presents a novel post-hoc regularization technique for tree-based models, called Hierarchical Shrinkage. Our main goal is to confirm the claims that it substantially increases the predictive performance of both decision trees and random forests, that it is faster than other regularization terhniques and that it makes the interpretation of random forests simpler.

Methodology — To reproduce the results we used the open-source implementation of Hierarchical Shrinkage, provided by the authors on GitHub in the Python package, and their code for obtaining data sets. We tested their implementation by manually calculating for different data sets and tree sizes, and compared provided data sets with the ones found online to verify their validity. We coded the experiments from scratch and to the best of our abilities as the authors describe them.

Results — We managed to reproduce most of the results the authors get. The method works well and most of the claims are supported. The method does increase the predictive performance of tree-based models most of the time, but not always. When compared to other regularization techniques the Hierarchical Shrinkage outperforms them when used on decision trees, but not when used on random forests. Since the method is applied after learning, it is extremely fast. And it does simplify decision boundaries for random forests, making them easier to interpret.

What was easy — The use of the official code for Hierarchical Shrinkage was straightforward and used the same function naming convention as other machine learning libraries. The function for acquiring already clean and pre-processed data sets saved a lot of time.

What was difficult — The authors also provided the code for their experiments in a separate library, but the code did not run out of the box and we had no success reproducing the results with it. The code was inconsistent with the paper methodology. We had the most problems with hyperparameter tuning. The authors did not specify how they tunned the hyperparameters for the used RF regularizers.

Communication with original authors — We did not contact the authors of the original paper.

Copyright © 2023 Anonymous, released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to ()
The authors have declared that no competing interests exists.
Code is available at https://anonymous.4open.science/r/hierarchicalshrinkage-re.

1 Introduction

Tree-based models (TBM) are commonly used in machine learning. The two most popular TBM are without doubt decision trees (CART) [1] and random forests (RF) [2]. The prior due to its simplicity is used to explain more complex model decisions, while the latter due to its robustness is used as a baseline for other models. Both models usually use the leaf sample average to make predictions. Agarwal et al [3] argue that this approach is not necessarily optimal and propose a new post-hoc algorithm called Hierarchical Shrinkage (HS) where models instead make predictions based on the weighted average of the mean responses over the leaf and each of its ancestors.
The authors first derive HS and compare it to other similar approaches: CART with cost-complexity pruning, C4.5 [4], and GOSDT [5]. The authors evaluate HS on both CART and RF models, across both classification and regression each containing eight commonly used datasets. In each test-case authors compare the TBM with HS and other regularization methods to assess whether HS improves the predictive power of the two most popular TBM and how it compares to other regularization algorithms in terms of speed and predictive improvement.
In our analysis, we used the Python library provided by Agarwal et al [3] for the HS, while for other algorithms we used the libraries and models cited in the paper. Our analysis mainly focused on statistically quantifying whether and by how much HS improves the predictive power of CART and RF relative to other regularization methods.

2 Scope of reproducibility

The authors of the HS method make many claims about improving predictive performance over other methods and making more robust and simpler explanations. In this paper, we focused on the four claims we deemed most important:

- **Claim 1: HS increases the predictive power of TBM.**
  The authors claim that the use of HS can improve the predictive performance of TBM, such as CART and RF.
- **Claim 2: HS is better than other regularization algorithms for TBM.**
  The authors claim that TBM with HS applied performs better than other regularization methods for TBM.
- **Claim 3: HS is faster than other regularization algorithms for TBM.**
  The authors claim that HS is faster than other regularization methods for TBM because it does not make structural modifications to the TBM.
- **Claim 4: HS leads to more intuitive and robust explanations of RF.**
  The authors claim that HS removes sampling artefacts and thus simplifies decision boundaries for RF and stabilizes feature importance, which makes it easier to interpret interactions in the model.

For each claim, we decided to use specific metrics to evaluate its accuracy. We used metrics such as AUC/ROC and R^2, and when comparing different methods, we used likelihood.

3 Methodology

We used the authors' code, provided in Python library imodels[1], for the Hierarchical Shrinkage algorithm. We also used one of their functions to load all of the clean datasets the authors use. We coded all of the experiments from scratch in Python 3.10.

[1] https://github.com/csinva/imodels

3.1 Model descriptions

HS algorithm modifies the prediction values of the nodes for a TBM. HS transforms the prediction function of the TBM to

$$\hat{f}_\lambda(x) = \mathbb{E}_{t_0}\{y\} + \sum_{l=1}^{L} \frac{\mathbb{E}_{t_l}\{y\} - \mathbb{E}_{t_{l-1}}\{y\}}{1 + \frac{\lambda}{N(t_{l-1})}} \tag{1}$$

where x is a query point, t_l denotes a node in the root-to-leaf path, $\mathbb{E}_{t_l}\{y\}$ is the expected value of target variable at node t_l, $N(t_l)$ is the number of instances at node t_l, and λ is a hyperparameter, either specified by the user or determined via cross-validation. The function can be seen as a weighted sum where more weight is given to the nodes that are closer to the root.
We used the CART algorithm to build decision trees and random forests. For specific claims, we used the C4.5 and GOSDT algorithms.

3.2 Datasets

We used the same datasets as the authors. The details can be found in Appendix A. We downloaded all of our datasets using the get_clean_dataset() function from the imodels library. The data in the datasets were cleaned by the authors, therefore by using these datasets we ensured that we used the same pre-processing methods as the authors. We also checked each dataset, searched for them online in public databases, and compared them to the ones the authors provide. We were able to find all of the datasets and ensured that the ones we found are the same ones the authors provide. The only dataset we were unable to verify was German credit. The dataset we found has the same features, number of instances, and target variable distribution, but some of the features have slightly different value distributions. Despite this discrepancy, we proceeded to use the dataset, provided by the authors.
We used $2/3$ of our data for training and $1/3$ for testing. For tunning, we used 3-fold cross-validation (CV) using only samples from the training set. All splits were done randomly (for classification tasks we used stratified splits).

3.3 Hyperparameters

For each experiment we performed several random splits; for each of the splits we found the best parameters with hyperparameter search (HPS). We always used the test metric as our objective function for HPS. In experiments, where the hyperparameter space was specified by the authors we used grid search (GS), while in instances when it was not we used bayesian optimization (BO) over a range of reasonable values. We performed GS manually and we used the gp_minimize() function from scikit-optimize for BO[2]. We selected the HS regularization parameter with GS from the set $\lambda_{HS} \in \{0.1, 1, 10, 25, 50, 100\}$. The same set of parameters was used in all instances where we applied HS post-hoc. Additionally, when comparing HS-RF with other regularization methods we used BO to select the optimal maximum depth of our RF trees from the range $d_{max} \in (1, 30)$, as well as the best fraction of features used when splitting a RF tree from the range $mtry \in (0.1, 1.0)$. These two post-hoc regularization methods were applied separately following the original paper. For the cost-complexity prunning (CCP) α parameter we used the approach from the original paper where α is chosen for each CV fold from the α returned by scikit-learn.cost_complexity_pruning_path such that the number of leaves in the prunned DT is as close as possible to the specified number of leaves.

[2]For BO we used expected improvement as the evaluation function. We conducted 15 trials each with 5 initial starting points. The noise in our kernel was set to $\sigma = 0.01$. We did not change other hyperparameters from their default values.

3.4 Experimental setup and code

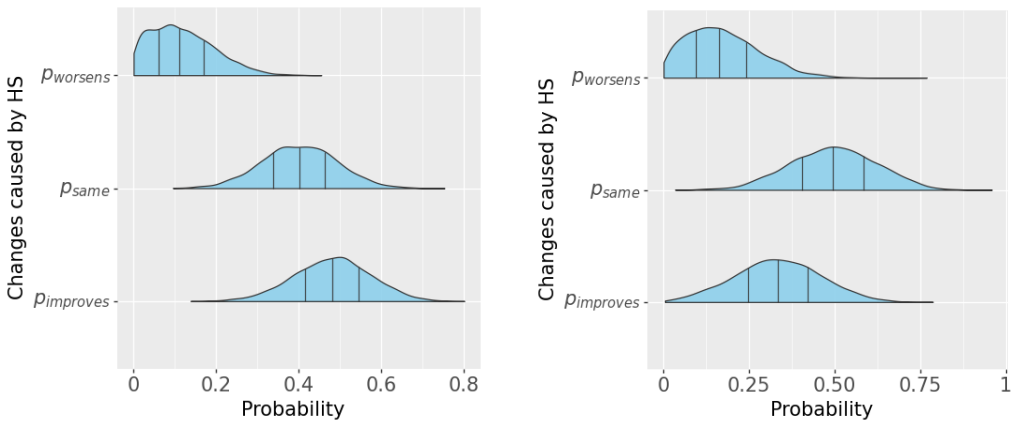
To perform our experiments we used the authors' code from their Python package imodels for applying HS and reading datasets and our code. We used Python 3.10 and libraries NumPy, pandas, scikit-learn, matpotlib, plotnine, and others. We structured our experiments into Jupyter notebooks, one notebook per claim.
We also tested the validity of their HS implementation by building deterministic trees with CART and manually calculating the application of HS. The tests are in folder tests. One tests the HS on regressive tasks and the other on classification tasks.
The metrics the authors and we use are the area under the curve (AUC/ROC) for classification and R^2 for regression. We take both from the scikit-learn library. For the improvement comparison, we used the region of practical equivalence (ROPE) of 0.005.

3.5 Computational requirements

We ran our experiments on two computers with the following two CPUs: Intel(R) Core(TM) i7-9750H CPU @2.60GHz (CPU_1) and AMD Ryzen 5 5600X 6-Core (CPU_2). Comparing CART with HS-CART for classification took 37 minutes (100 repeats) while performing a similar experiment comparing CCP (100 repeats) with HS took 4 hours. Measurements were done in wall time. Comparing RF with other regularization methods on classification datasets took roughly 4.5 hours. For DT cpu and wall time are almost identical, while for RF the cpu time is roughly 50% larger than wall time.

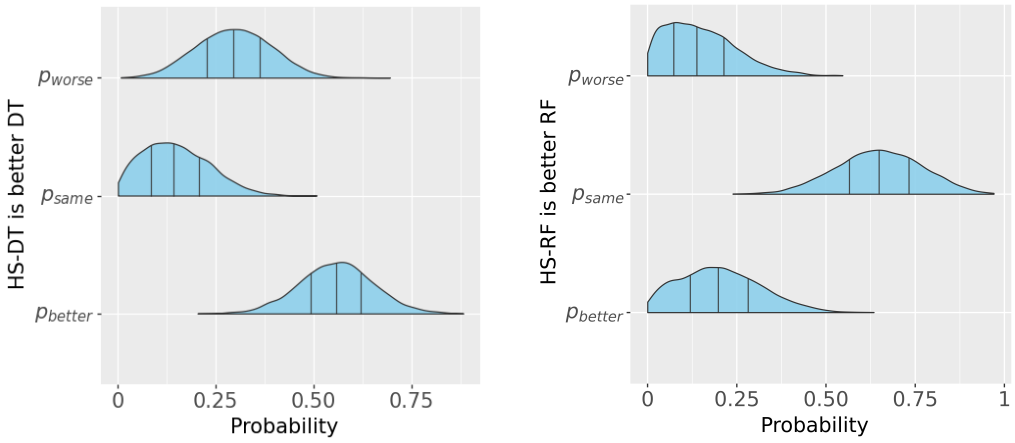
4 Results

Our results show that HS improves DT scores and is one of the best forms of regularization in comparison to other DT regularization methods both in terms of predictive power as well as in terms of speed. Our results do not indicate that HS is better for RF than other regularization methods in terms of predictive power, however, they do show that HS is significantly faster than other regularization methods and is easier to interpret.



(a) Likelihood of improvement for DT. (b) Likelihood of improvement for RF.

Figure 1. This set of images shows the likelihood that HS improves a TBM score. For DT, the likelihood of improvement is the highest, while for RF the likelihood of no change is the highest.



(a) Likelihood that HS-DT is better than DT.

(b) Likelihood that HS-RF is better than RF.

Figure 2. This set of images shows the likelihood that a TBM with HS is better than one without HS. There is a significant likelihood of HS-DT being better than DT, while there is no significant likelihood that HS-RF will be better than RF.

4.1 Claim 1: HS increases the predictive power of TBM.

The authors claim that applying HS to TBM significantly improves their prediction performance. They test this by applying HS to DT and RF across 16 different datasets; authors vary the maximum number of leaves for DT and the maximum number of trees for RF to test how HS improves TBM when the complexity of the models changes. The authors evaluate the results using several metrics, though they mainly focus on AUC/ROC for classification and R^2 for regression.

We split the author's claim that HS improves the predictive power of TBM into the likelihood that HS improves the model it is applied to (Fig. 1) and the likelihood that HS applied to a TBM on average gives better results (Fig. 2). The authors mix these two claims thereby summing together the variance of TBM with the variance of the improvement resulting from HS, as such their results do not conclusively corroborate either of the claims.

To analyze whether HS improved a specific TBM we created a categorical variable that represented whether HS improved the score of the TBM. We then fit a hierarchical categorical model[3], where the first level represented the dataset, and the category in the second level was whether applying HS improves our TBM score.

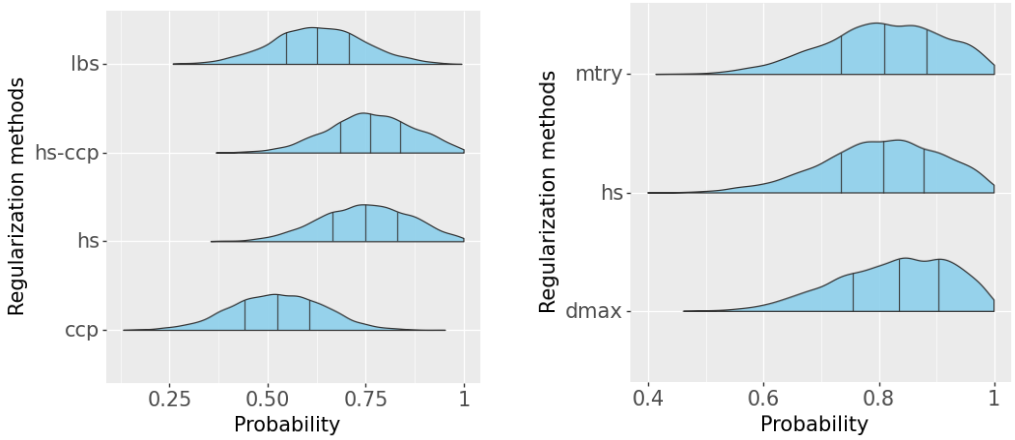
To analyze whether HS-TBM was on average better than TBM, we split our test scores into groups where each group was comprised of test scores from runs on the same dataset and the same number of leaves or trees, then for each model we randomly with replacement sampled 1000 instances using each regularization method and saved how HS-TBM fared in comparison to TBM. We then fit a hierarchical Categorical model[3], where the first level represented the dataset, and the category in the second level was how HS-TBM fared in comparison to TBM.

Our results (Fig. 1) confirm the sub-claim that by applying HS to the DT the score is likely to increase, however, this does not apply to RF with the likelihood of the score not changing being larger than 50%. Our results (Fig. 2) confirm the sub-claim that the likelihood of an HS-DT score is on average significantly better than a DT score (Fig. 2a), however, they do not confirm the sub-claim that HS-RF is on average better than RF (Fig. 2b), with a likelihood of over 60% that there is no significant difference between HS-RF and RF. More detailed results are in Appendix B.

[3] We used the same normal prior $p_i[d] \sim N(1/n, 0.5)$, where $p_i[d]$ is the probability of the event being in category i, n is the number of categories and d is the samples dataset.

4.2 Claim 2: HS is better than other regularization algorithms for TBM.

The authors claim that DT with HS regularization achieves better results than DT with CCP or LBS regularization and that RF with HS regularization achieves better results than RF with $mtry$ or d_{max} hyperparameter tunning. The authors use the same evaluation metrics as in Claim 1. For each dataset, the authors train all three TBM ten times and then evaluate it with AUC/ROC or R^2.



(a) Likelihood of regularized DT achieving the best score

(b) Likelihood of regularized RF achieving the best score

Figure 3. This set of images shows the likelihood that TBM regularized with a certain regularization method achieves the best score. For DT HS is the best regularizer, significantly outperforming CCP and LBS. For RF all three method scores are almost indistinguishable.

We split our test scores into groups where each group was comprised of test scores from runs on the same dataset and the same number of leaves or trees, then for each model we randomly with replacement sampled 1000 instances using each regularization method and saved whether the method achieved the best results. Finally, we fit a hierarchical Bernoulli model[3], where the first level represented the dataset, and the category in the second level was whether the model was the best or not.
Our results (Fig. 3) confirm the author's claim that HS is a better regularizer than CCP or LBS, with both HS-CCP and HS performing similarly indicating we are unlikely to significantly improve our results by combining HS and CCP. Our results however do not confirm that HS is a better regularizer for RF compared to $mtry$ or d_{max}, with both methods performing similarly to HS. More detailed results are in Appendix C.

4.3 Claim 3: HS is faster than other post hoc regularization algorithms for TBM.

The authors claim that HS is faster than other post-hoc regularization algorithms for TBM. The authors did not specify what experiments they conducted to verify this claim, however, we assumed the authors measured the wall and/or CPU time necessary to tune, train and test each model.
To check this claim, we measured both the CPU and wall time for the tunning, training and testing phase of each regularization algorithm for each dataset. We summed the three phases together and calculated the difference between the mean times of HS and other regularization methods (Tab. 1).
Our results confirm the claim that HS is significantly faster in CPU time in comparison to all other regularization methods. While HS is also better in wall time the improvement is less pronounced. The only instance where HS is slower than all other regularization methods is in the case of the Recidivism dataset. Considering this is true when com-

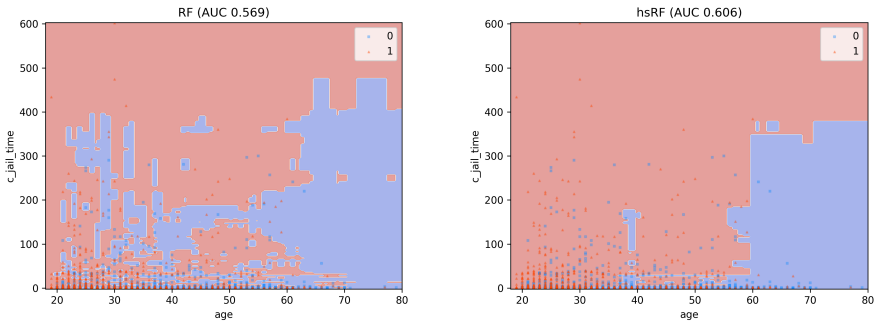
paring HS to both other regularization methods we consider this difference significant. Comparison of time complexity of different DT regularizers is in Appendix D.

	base	overtime		base	overtime	
dataset	hs wall	$mtry$ wall	d_{max} wall	hs cpu	$mtry$ cpu	d_{max} cpu
Heart	3	2 ± 1	2 ± 1	3	11 ± 1	11 ± 1
Breast cancer	4	2 ± 1	2 ± 1	4	10 ± 1	11 ± 1
Haberman	4	2 ± 1	2 ± 1	4	11 ± 1	11 ± 1
Ionosphere	3	3 ± 1	4 ± 1	3	12 ± 1	13 ± 1
Diabetes	5	1 ± 1	2 ± 1	5	10 ± 1	10 ± 1
German credit	6	0 ± 1	1 ± 1	6	9 ± 1	10 ± 1
Juvenile	12	-2 ± 5	23 ± 2	12	7 ± 5	31 ± 2
Recidivism	43	-33 ± 6	-24 ± 6	43	-24 ± 6	-16 ± 6

Table 1. Average execution overtime ($\pm$ stderr) in seconds of regularized RF relative to HS-RF for classification. Blue indicates HS-RF was faster, red indicates it was slower and grey indicates the difference in mean execution times was smaller than the standard error of the difference.

4.4 Claim 4: HS leads to more intuitive and robust explanations of RF.

The authors state that applying HS to RF removes sampling artefacts and simplifies decision boundaries. They test this by building an RF with 50 trees on only two of the most important features, determined by the mean decrease in impurity (MDI, Gini index). Our results (Fig. 4) show that the boundaries are smoother after applying HS. HS prevents overfitting and therefore smoothes the boundary between the target classes. With simpler boundaries the model predictions can be better explained, leaving much less uncertainty. The boundaries for all of the classification datasets are in Appendix E.1.



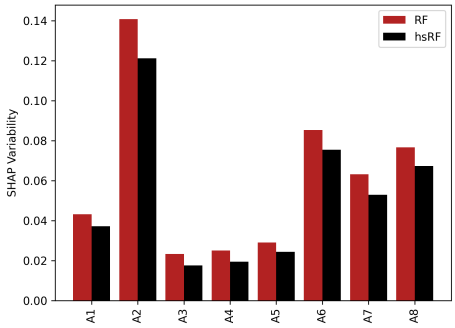
(a) Recidivism RF (AUC 0.569)

(b) Recidivism hsRF (AUC 0.606)

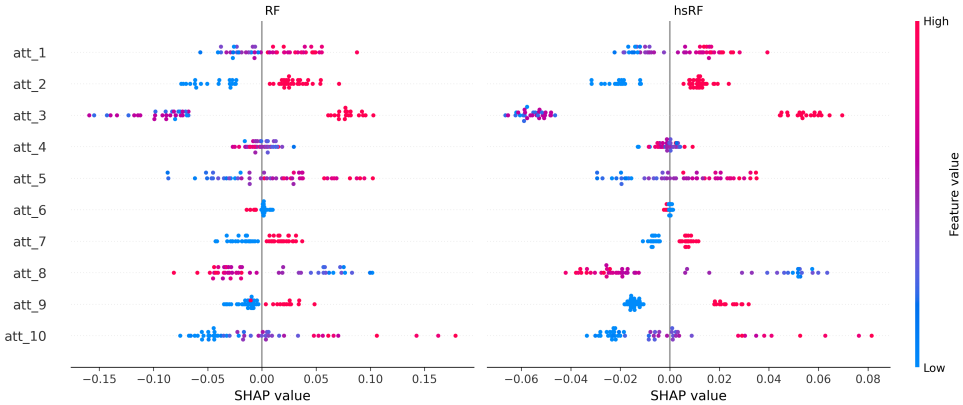
Figure 4. Applying HS to RF simplifies boundaries. We have the decision boundary before (4a) and after (4b) applying HS on the Recidivism dataset on the two most important features, age and c_jail_time.

We found two things that did not agree with the author's results. The change in AUC after applying HS seemed to be very much dependent on the tree structure and did not always result in higher scores. After applying HS the AUC changed unpredictably, either improving or worsening the performance. Another thing we found different was the top two features, determined by MDI. We found the same two most important features on

two datasets (Diabetes and Breast cancer) out of the eight, and for one (Heart) the top two features were completely different.
We applied the SHAP values to the resulting RF, which is used for model-agnostic explanations. The authors claim that HS improves stability and makes tighter clusters of SHAP values. Our results (Fig. 5) show that HS lowers the variability of SHAP values, which makes them more stable. It creates clusters in SHAP values, meaning they can be more easily interpreted. One thing the authors do not discuss is the smaller SHAP values - HS does make interpretation more stable, but at the same time lowers the importance of all features. This, however, does not influence the feature importance for classification tasks as only the relative relations matter. The results for all other datasets are in Appendix E.2.



(a) SHAP variance on Diabetes dataset.



(b) SHAP values for Heart dataset.

Figure 5. SHAP values before and after applying HS. In the upper figure (5a) we have the variance reduction for the Diabetes dataset. In the lower figure (5b) we have the SHAP values for the Heart dataset. We can see that after applying HS (right) the SHAP values are clustered, which makes features easier to interpret.

5 Discussion

In this report, we managed to show that HS, as a regularization technique, consistently improves DT scores. HS was also on average the best regularizer for DT. For RF, HS did not perform better than other regularization methods and did not consistently improve RF, however, it was significantly faster compared to other regularization methods. We hypothesize that the divergence between our observed (Appendix B, C) and the original paper scores are likely because the differences between our TBM scores were smaller. We assume that for $mtry$ and d_{max} the differences might have been caused by using a

different HPS space, possibly indicating a poor choice of HPS space on the author's part, while in the instance of RF our differences might be pure chance since the authors used only 10 samples for their experiments, therefore some variance in score distributions is expected. The decision boundaries for RF were less fragmented after applying HS, which made them easier to interpret. HS also lowered SHAP value variability, and as a result, HS-RF generated more stable explanations. Based on our results we conclude that HS is one of the best (evaluated) regularization methods for DT in terms of all three aspects (score, speed, interpretability). For RF, HS as a regularization method only stood out in terms of speed, scoring similarly to other regularization methods. We also note that our results did show that HS improved interpretability, however, no comparison was done with other regularization methods.

5.1 What was easy

It was easy to use HS and other TBM that were implemented in the imodels library. The library used the same naming conventions as the scikit-learn library, implementing most of the standard fit(), predict() and predict_proba() functions. Furthermore, the imodels library provides a get_clean_datasets() function from which we were able to download already preprocessed datasets, allowing us to focus more of our effort on the reproduction.

5.2 What was difficult

Parts that required changes to the code or that were not present in the imodels library were generally problematic to implement. While the article did link to the paper repository the tutorial notebooks did not run out of the box and required significant modification to run. One methodological problem was the usage of number of leaves during evaluation in spite of the fact that a number of leaves was not a hyperparameter in any of the methods implemented in either imodels or scikit-learn. We concluded that the authors most likely specified the maximum number of leaves a tree could have, but we were unable to verify whether the authors used the max. number of leaves or the number of non-leaf nodes (a model attribute implemented for most TBM in the imodels library) for evaluation. This was because the code in the notebooks used the latter for evaluation, while the original paper claimed to use the prior.
Another problem we encountered was that the authors did not specify how they performed hyperparameter search for methods such as $mtry$ and d_{max}.

5.3 Communication with original authors

We did not contact the original authors. Most of the choices in the original paper were well documented and we were generally able to fill in the gaps for missing specifications based on the contents of the corresponding claim. Additionally, we also considered not contacting the authors to have specific benefits such as using improved evaluation methodology or using different HPS techniques, which allowed us to test the robustness of the author's claims to small alterations in the proposed methodology. By taking this approach while our reproducibility might have possibly suffered it enabled us to expand on the authors' results in certain areas.

References

1. L. Breiman, J. Friedman, C. Stone, and R. Olshen. **Classification and Regression Trees**. Taylor & Francis, 1984.
2. L. Breiman. "Random forests." In: **Machine learning** 45.1 (2001), pp. 5–32.

3. A. Agarwal, Y. S. Tan, O. Ronen, C. Singh, and B. Yu. "Hierarchical Shrinkage: Improving the accuracy and interpretability of tree-based models." In: **Proceedings of the 39th International Conference on Machine Learning**. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 111–135.
4. J. R. Quinlan. **C4. 5: programs for machine learning**. Elsevier, 2014.
5. J. Lin, C. Zhong, D. Hu, C. Rudin, and M. Seltzer. "Generalized and Scalable Optimal Sparse Decision Trees." In: **Proceedings of the 37th International Conference on Machine Learning**. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 6150–6160.

Appendix

A Datasets

Here we list all of the datasets with their corresponding information (Tab. 2). We used eight classification and eight regression datasets. For each dataset, we list the number of samples, the number of features, the distribution of the target variable, and a link to the dataset.

	dataset	Samples	Features	Target	Link
Classification	Heart	270	15	44.4%	UCI ML repository
	Breast cancer	286	9	29.2%	UCI ML repository
	Haberman	306	3	73.5%	UCI ML repository
	Ionosphere	351	34	64.1%	UCI ML repository
	Diabetes	768	8	34.9%	Kaggle
	German credit	1000	20	70.0%	UCI ML repository
	Juvenile	3640	286	13.4%	nacjd
	Recidivism	6172	20	48.4%	ProPublica Data Store
Regression	Friedman1	200	10	14.25 ± 5.01	scikit-learn
	Friedman3	200	4	1.33 ± 0.30	scikit-learn
	Diabetes	442	10	152.1 ± 77.1	scikit-learn
	Geographical music	1059	117	0.02 ± 1.00	Penn ML Benchmarks
	Red wine	1599	11	5.63 ± 0.81	UCI ML repository
	Abalone	4177	8	9.93 ± 3.22	UCI ML repository
	Satellite image	6435	36	3.67 ± 2.21	Penn ML Benchmarks
	CA housing	20640	8	2.07 ± 1.15	Kaggle

Table 2. Datasets used by the authors throughout their paper. The Target column displays the percentage of positive cases for classification tasks, and mean response value with standard deviation for regression tasks.

B Claim 1: HS increases the predictive power of TBM.

Based on Fig 9 we can see that our results are similar in terms of mean score to the original paper, the only consistent difference being the larger variance. This explains the observed disparity between ours and the author's results since while HS-DT performs better than DT, the difference is not statistically significant[4] due to the larger variance.
We further analyzed the likelihood that the top scores of HS-DT were better than the top DT scores (Fig. 7a) and found this to be the case in nearly all instances with improvements being less pronounced for regression than for classification. This might indicate that HS influences different evaluation metrics differently.
When checking the likelihood of improvement (Fig. 7b) we can see that the likelihood that applying HS to DT improves the scores or does not significantly worsen them is almost 100% indicating that HS is indeed a valid form of regularization for DT.
Finally, we checked how HS improved DT when the DT was trained using the default number of leaves[5]. This is $n/3$ and $\sqrt{n}$ for regression and classification, respectively. Here it is apparent that there are no improvements, likely due to the fact that the defaults build small trees that cannot be regularized significantly, therefore users should not expect improvements if they use HS while using default parameters.

[4]We considered differences statistically significant if they differed by one standard deviation.
[5]Missing bars indicate that DT built less than five trees near the default number of trees making it so that our conclusions would not have enough significance for those datasets.

When comparing top scores of HS-RF and RF (Fig. 8a) it appears that the likelihood that we can generate a top score using RF is similar to the likelihood that we can generate a top score using HS-RF indicating that HS does not significantly improve RF scores. This conclusion is enforced by the comparison of RF and RF after HS was applied to it (Fig. 8b) indicating that while some improvements were still visible these improvements were much smaller and much less consistent across datasets. We conclude this is likely due to the regularizing effect of using ensembles which decreases the potential improvements offered by HS.

C Claim 2: HS is better than other regularization algorithms for TBM.

In this section, we investigate whether HS is better than other regularization algorithms for TBM. Based on our results we conclude that HS is not significantly better than other regularization methods. For DT HS-CCP is significantly better than CCP, but not significantly better than LBS (Fig. 9a & 9b[6]) as the later occasionally performs better and occasionally worse but always within one standard deviation of HS-CCP.
For RF (Fig. 9c) we can see that in most instances all methods perform similarly to one another, the exceptions being the breast-cancer and recidivism dataset. There is no consistency in how our scores differ from the original papers; for datasets like haberman they are higher, for breast-cancer they are lower and for heart they are roughly the same. Based on the above observation and the high variance of scores we conclude that neither our nor the author's results are necessarily conclusive and more samples would need to be generated in order to definitively show which method is better.

D Claim 3: HS is faster than other post-hoc regularization algorithms for TBM.

In this section, we investigate the execution time of HS in comparison to other regularization algorithms for DT. Based on our results we conclude that as the size of the dataset grows HS the differences in speed become more pronounced with HS becoming significantly faster for larger datasets. Additionally, it appears that especially for small datasets CCP is faster than HS. We also note that there appear to be no significant differences between cpu and wall time indicating that parallelization has no impact on the speed of regularization of DT.
We see that HS takes up a considerable amount of the overall execution time of HS-RF (Fig. 10b), therefore we conclude that using HS for regularization has substantial time costs albeit lower than that of other evaluated regularization methods. Additionally, the time costs appear to grow as the dataset size increases. The opposite appears to be the case for DT were the time cost decreases as the dataset grows indicating that HS can be used to regularize DT without significantly increasing the execution time of DT. Finally, we note that Fig. 10 only displays the relative wall time of HS-TBM, however this was done intentionally since relative wall and cpu times were the same up to two decimal places for both HS-TBM.

[6]We did not calculate the experiment for california-housing due to the large execution time of CCP on large datasets; To train one DT regularized with CCP it took approx 1.5 h of wall time (see Tab. 3)

	base	overtime	base	overtime
dataset	hs wall	ccp wall	hs cpu	ccp cpu
heart	0.019	-0.010 ± 0.000	0.019	-0.010 ± 0.000
breast-cancer	0.019	-0.009 ± 0.000	0.019	-0.009 ± 0.000
haberman	0.018	-0.008 ± 0.000	0.018	-0.008 ± 0.000
ionosphere	0.031	-0.002 ± 0.000	0.031	-0.002 ± 0.000
diabetes	0.027	0.042 ± 0.000	0.027	0.042 ± 0.000
german-credit	0.028	0.099 ± 0.001	0.028	0.099 ± 0.000
juvenile	0.149	2.372 ± 0.014	0.149	2.372 ± 0.014
recidivism	0.057	5.381 ± 0.012	0.057	5.381 ± 0.012
friedman1	0.013	0.059 ± 0.000	0.013	0.059 ± 0.000
friedman3	0.011	0.039 ± 0.000	0.011	0.039 ± 0.000
diabetes-regr	0.016	0.315 ± 0.001	0.016	0.315 ± 0.001
red-wine	0.034	0.927 ± 0.004	0.034	0.927 ± 0.004
abalone	0.054	16.031 ± 0.033	0.054	16.026 ± 0.033
satellite-image	0.224	11.108 ± 0.058	0.224	11.107 ± 0.058
california-housing	0.365	5149.957 ± 2.861	0.365	5149.957 ± 2.855

Table 3. Average execution overtime ($\pm$ standard deviation) in seconds of regularized DT relative to HS-DT for classification. Blue indicates HS-DT was faster, red indicates it was slower and grey indicates the difference in mean execution times was smaller than the standard error of the difference.

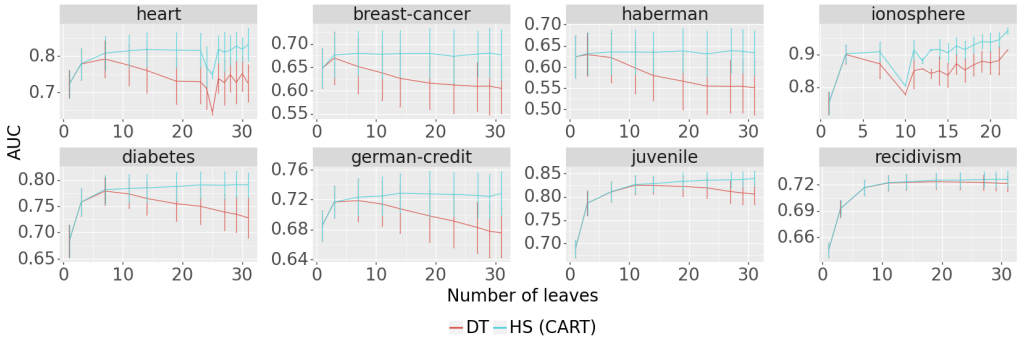
E Claim 4: HS leads to more intuitive and robust explanations of RF.

E.1 Decision boundaries

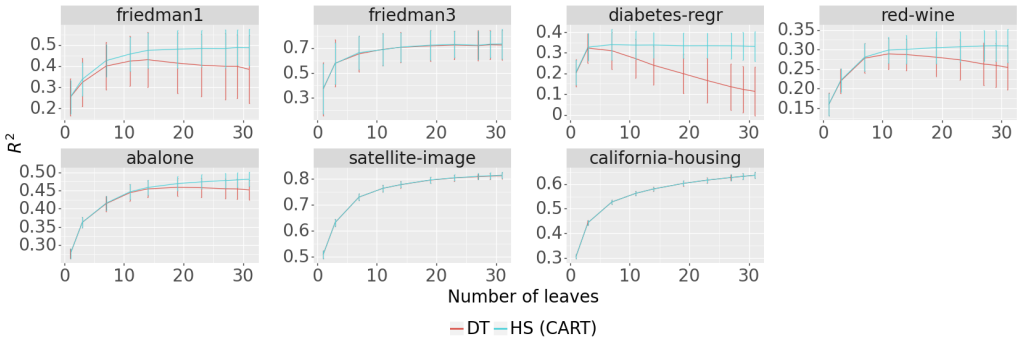
In this section we investigate how the HS influences the decision boundaries of classification problems as learned by RF. The two features were picked by best MDI. The comparison of boundaries before and after applying HS is on Fig. 11 and Fig. 12. In all of the cases the resulting decision boundary is simpler and less fragmented, which makes the model easier to interpret.

E.2 SHAP plots and variability

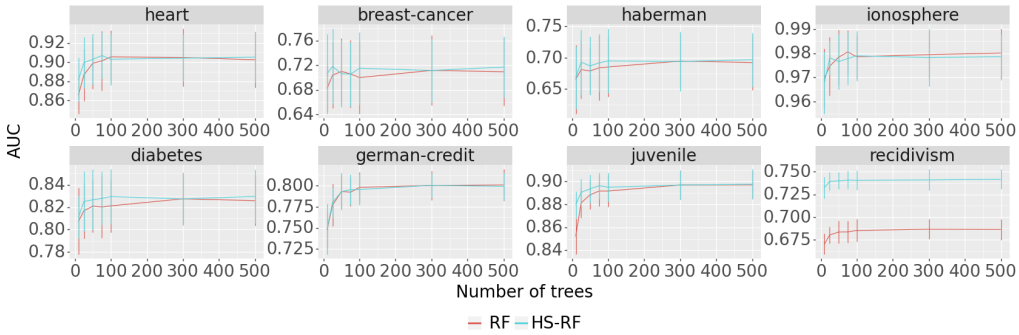
In this section we investigate how HS influences the SHAP value plots and their variability for RF with and without HS. The experiment was done by witholding 50 data points from a classification problem, build a RF with 2/3 of the remaining data, and evaluate SHAP values for each held out sample. This was repeated 100 times. We then averaged the variance for each feature across all held out samples.
Value plots for classification data sets are on Fig. 13 and Fig. 14. We can see that after applying HS the explanations for each feature have tighter clusters, and explanations are less noisy.
We showed that HS clusters the explanations together, which means it reduces the explanation variance. The comparison of the variability of SHAP interpretations for classification data sets can be seen on Fig. 15 and Fig. 16.



(a) DT AUC score before and after applying HS regularization.

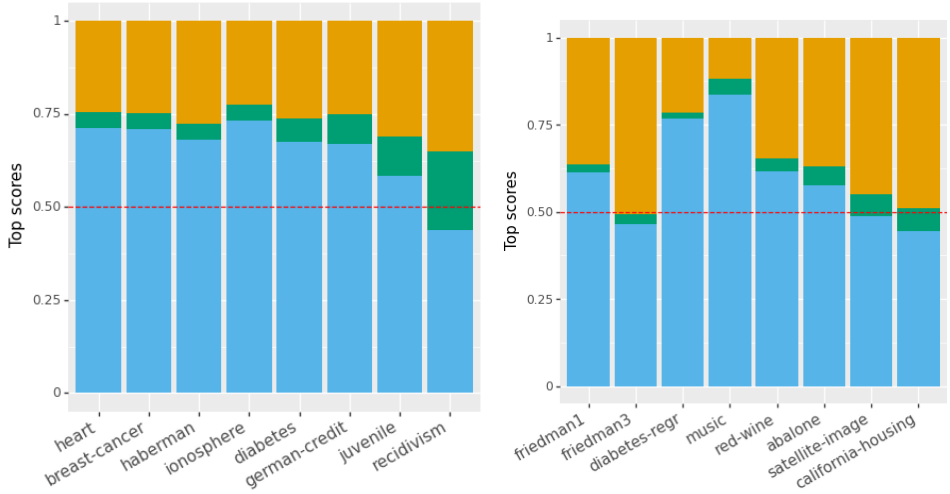


(b) DT R^2 score before and after applying HS regularization.

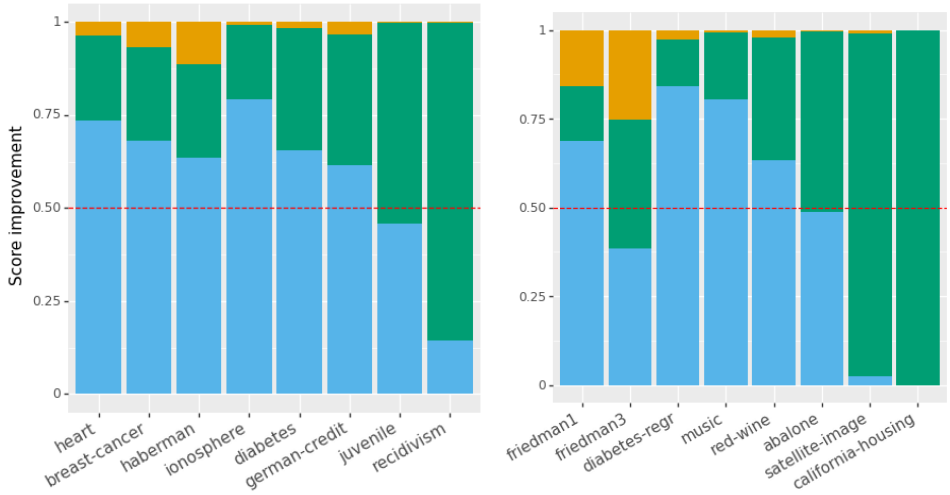


(c) RF AUC score before and after applying HS regularization.

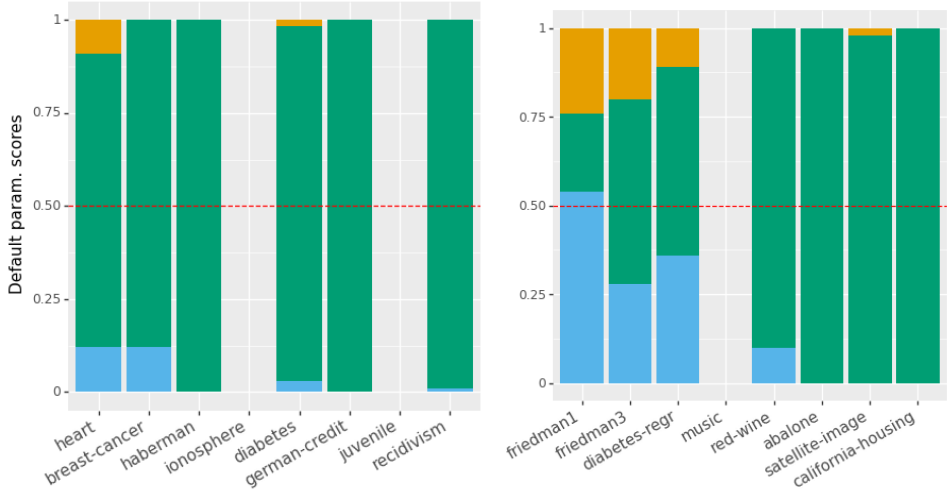
Figure 6. HS improves DT AUC and R^2 score for smaller datasets with the improvements decreasing as the dataset size grows, however for RF no such improvement is visible.



(a) Comparison of top scores between HS-DT and DT for each dataset.

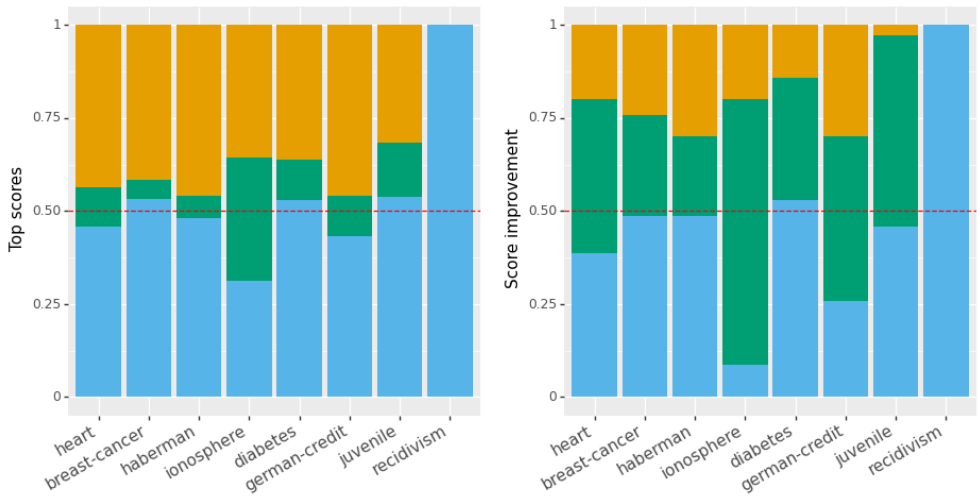


(b) Comparison of improvement in DT scores before and after HS regularization for each dataset.



(c) Comparison of scores between HS-DT and DT when using default hyperparameters for each dataset.

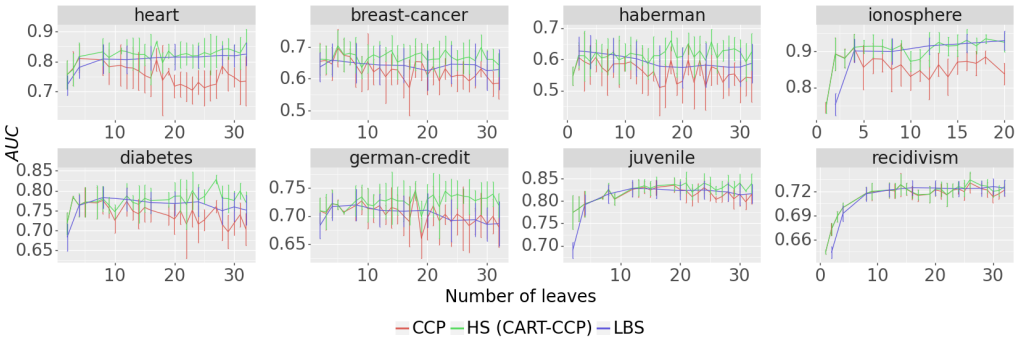
Figure 7. Each figure in our image set represents the likelihood that out of 1000 sampled pairs HS scored better (blue), both pairs scored the same (green) or DT scored better (orange).



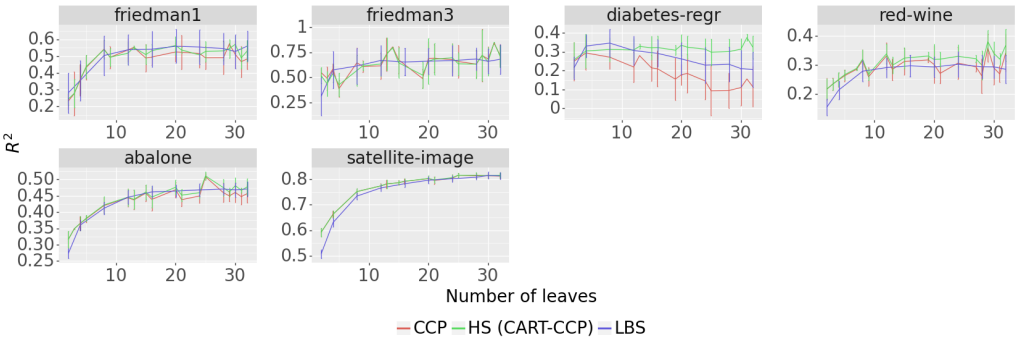
(a) Comparison of top scores of HS-RF and of RF for each dataset.

(b) Comparison of improvement in RF scores before and after HS regularization for each dataset.

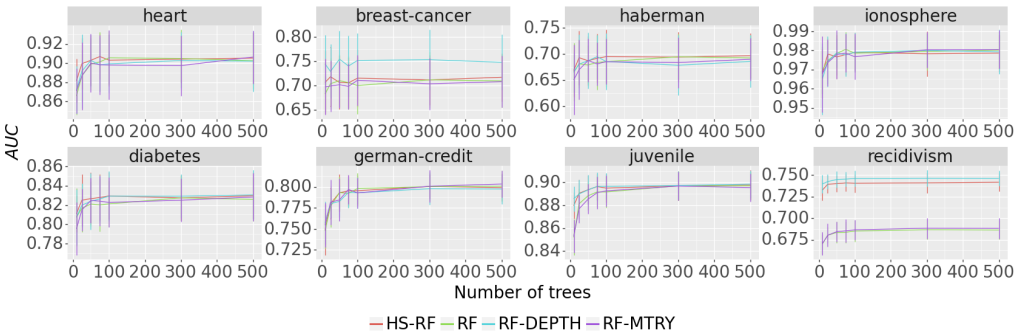
Figure 8. Each figure in our image set represents the likelihood that out of 1000 sampled pairs HS scored better (blue), both pairs scored the same (green) or RF scored better (orange).



(a) DT AUC score before and after applying HS regularization.

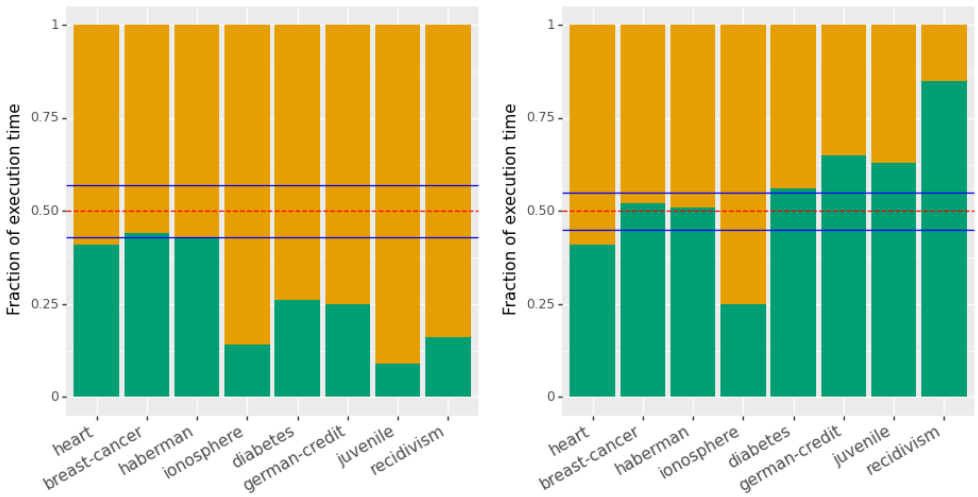


(b) DT R^2 score before and after applying HS regularization.



(c) RF AUC score before and after applying HS regularization.

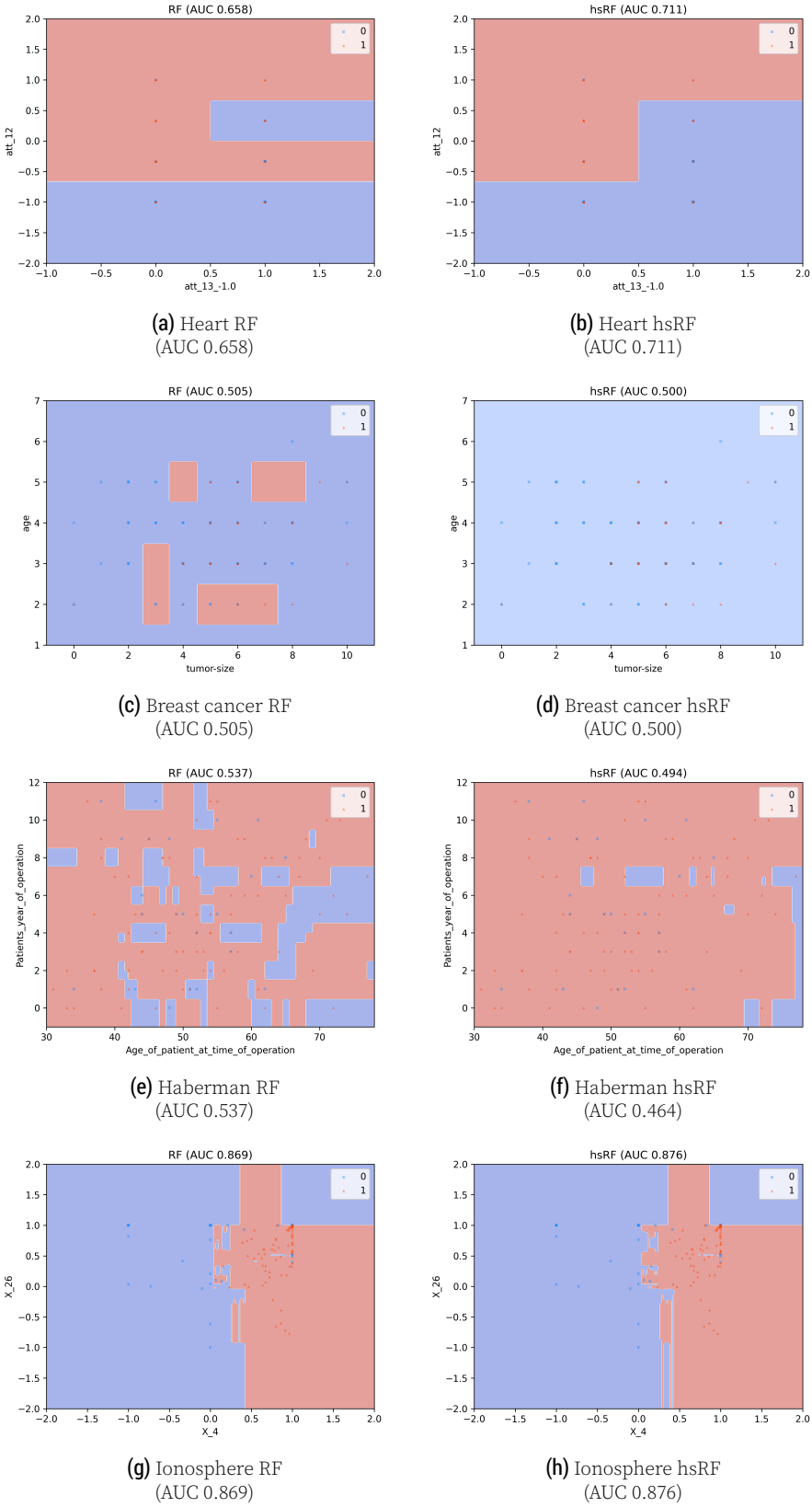
Figure 9. HS improves DT AUC and R^2 score for smaller datasets with the improvements decreasing as the dataset size grows, however for RF no such improvement is visible.



(a) Fraction of HS-DT execution time spent on DT (orange) and HS (green).

(b) Fraction of HS-RF execution time spent on RF (orange) and HS (green).

Figure 10. Dotted red line marks 50% and blue lines mark largest standard deviation across all datasets.



(a) Heart RF (AUC 0.658)

(b) Heart hsRF (AUC 0.711)

(c) Breast cancer RF (AUC 0.505)

(d) Breast cancer hsRF (AUC 0.500)

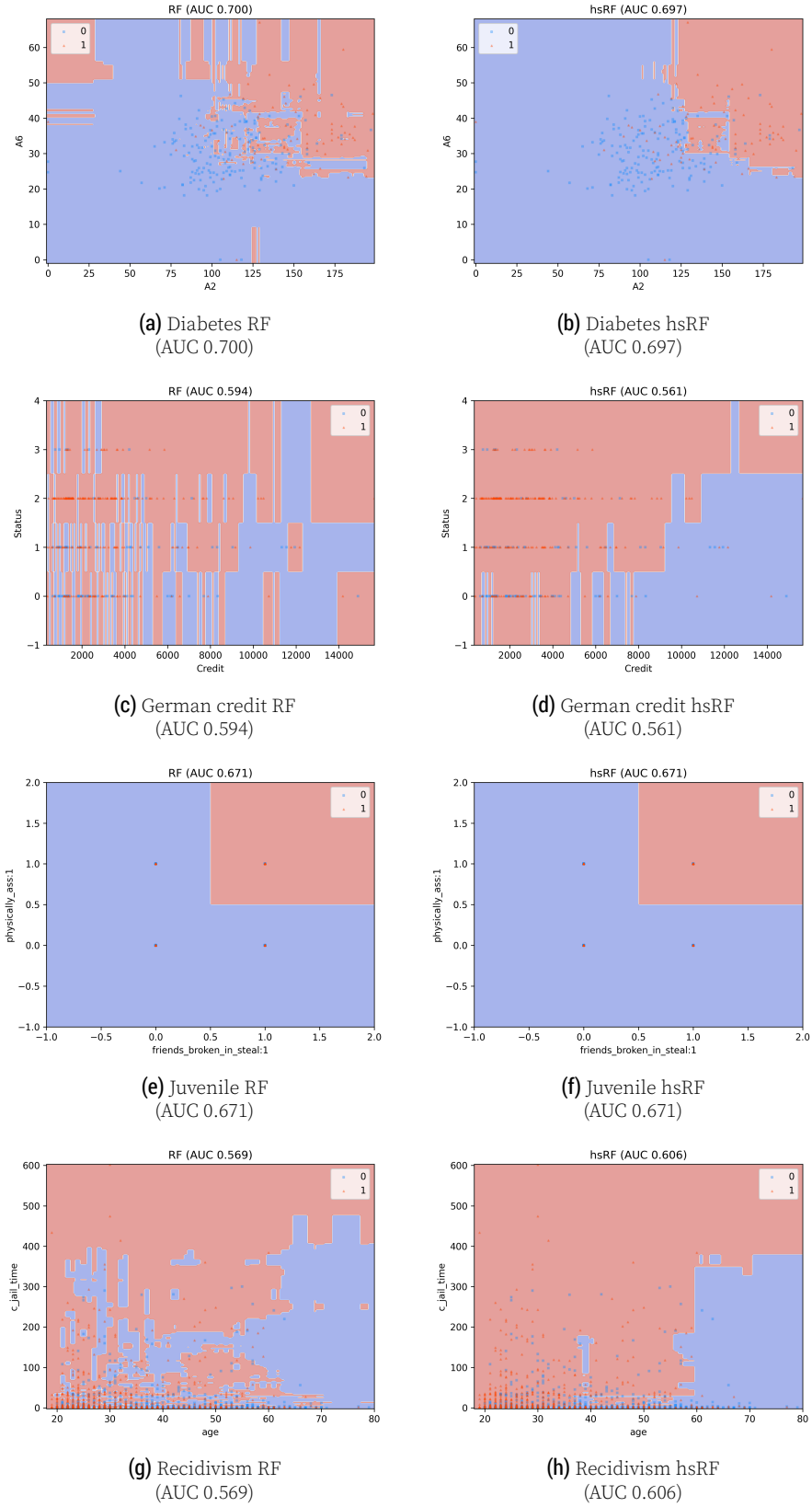
(e) Haberman RF (AUC 0.537)

(f) Haberman hsRF (AUC 0.464)

(g) Ionosphere RF (AUC 0.869)

(h) Ionosphere hsRF (AUC 0.876)

Figure 11. Comparison of decision boundaries for the first four classification data sets as learned by RF before and after applying HS.



(a) Diabetes RF (AUC 0.700)

(b) Diabetes hsRF (AUC 0.697)

(c) German credit RF (AUC 0.594)

(d) German credit hsRF (AUC 0.561)

(e) Juvenile RF (AUC 0.671)

(f) Juvenile hsRF (AUC 0.671)

(g) Recidivism RF (AUC 0.569)

(h) Recidivism hsRF (AUC 0.606)

Figure 12. Comparison of decision boundaries for the last four classification data sets as learned by RF before and after applying HS.

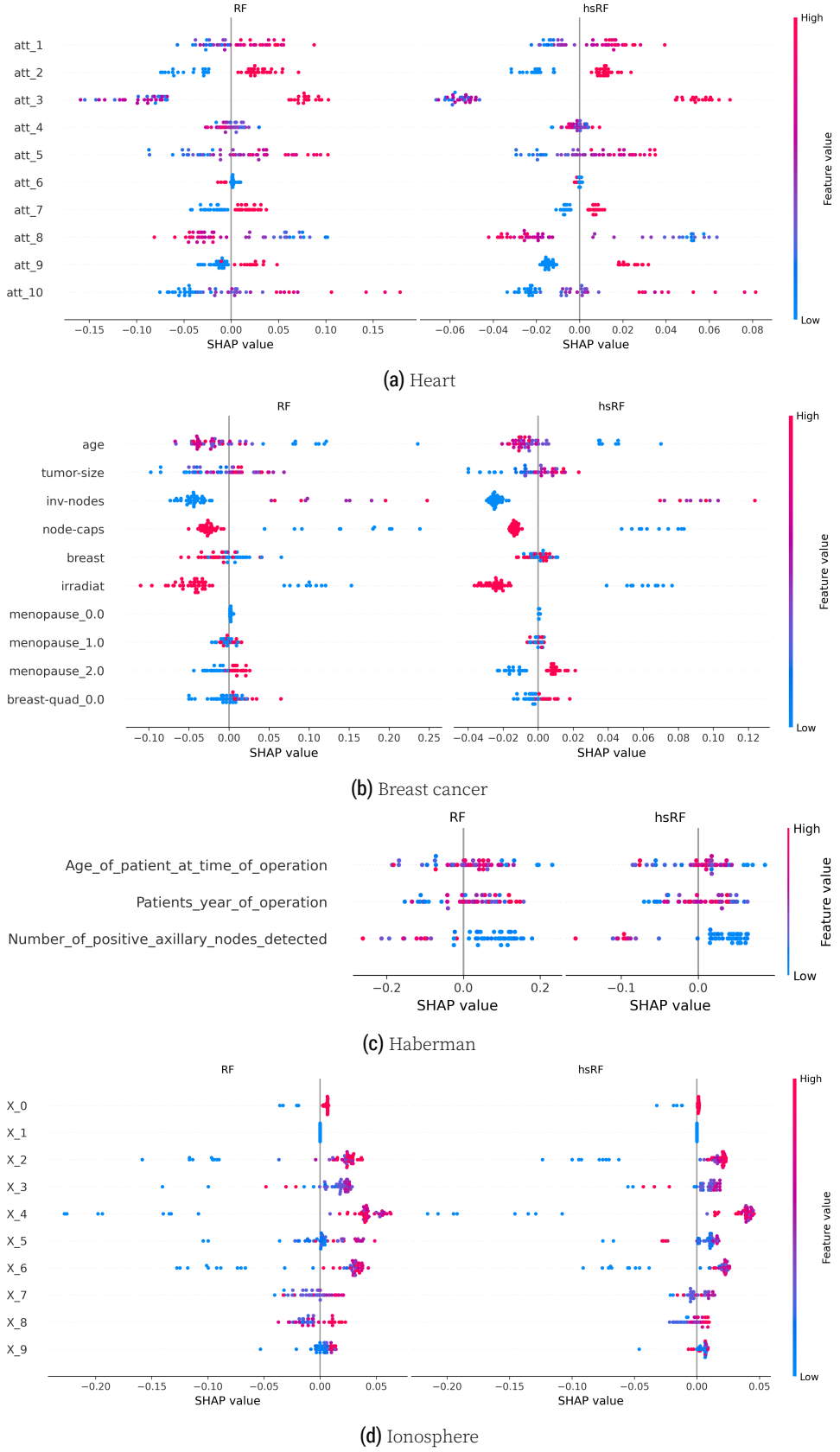
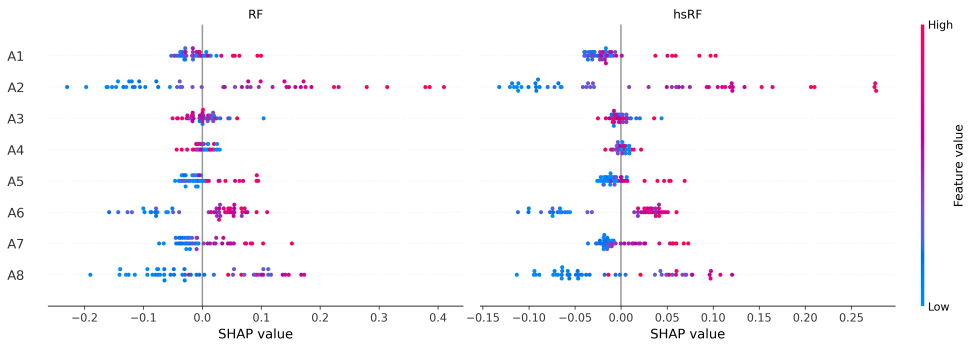
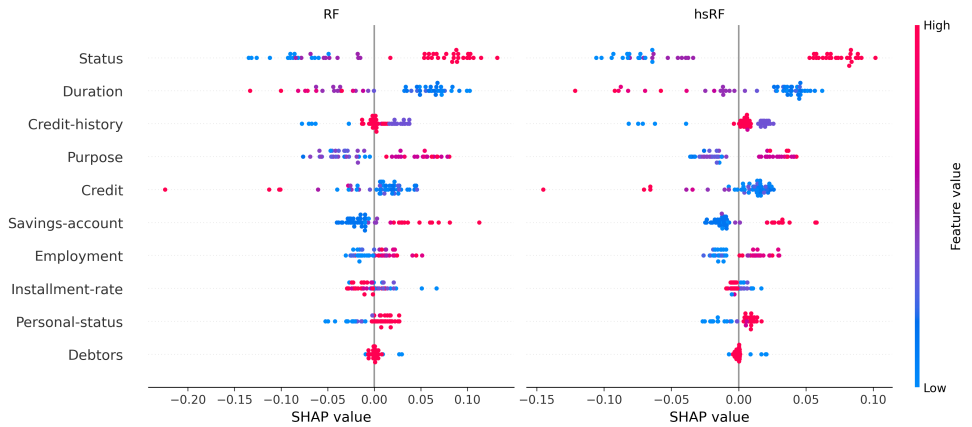


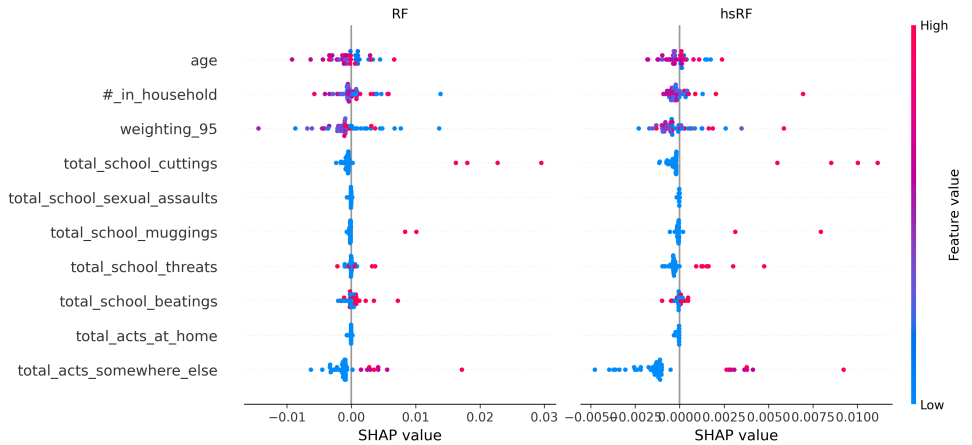
Figure 13. Comparison of SHAP value plots for the first four classification data sets before and after applying HS.



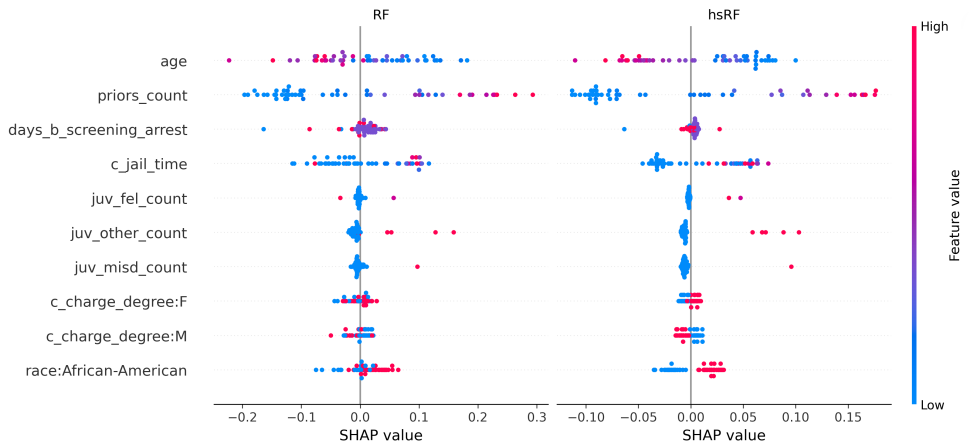
(a) Diabetes RF



(b) German credit



(c) Juvenile



(d) Recidivism

Figure 14. Comparison of SHAP value plots for the last four classification data sets before and after applying HS.

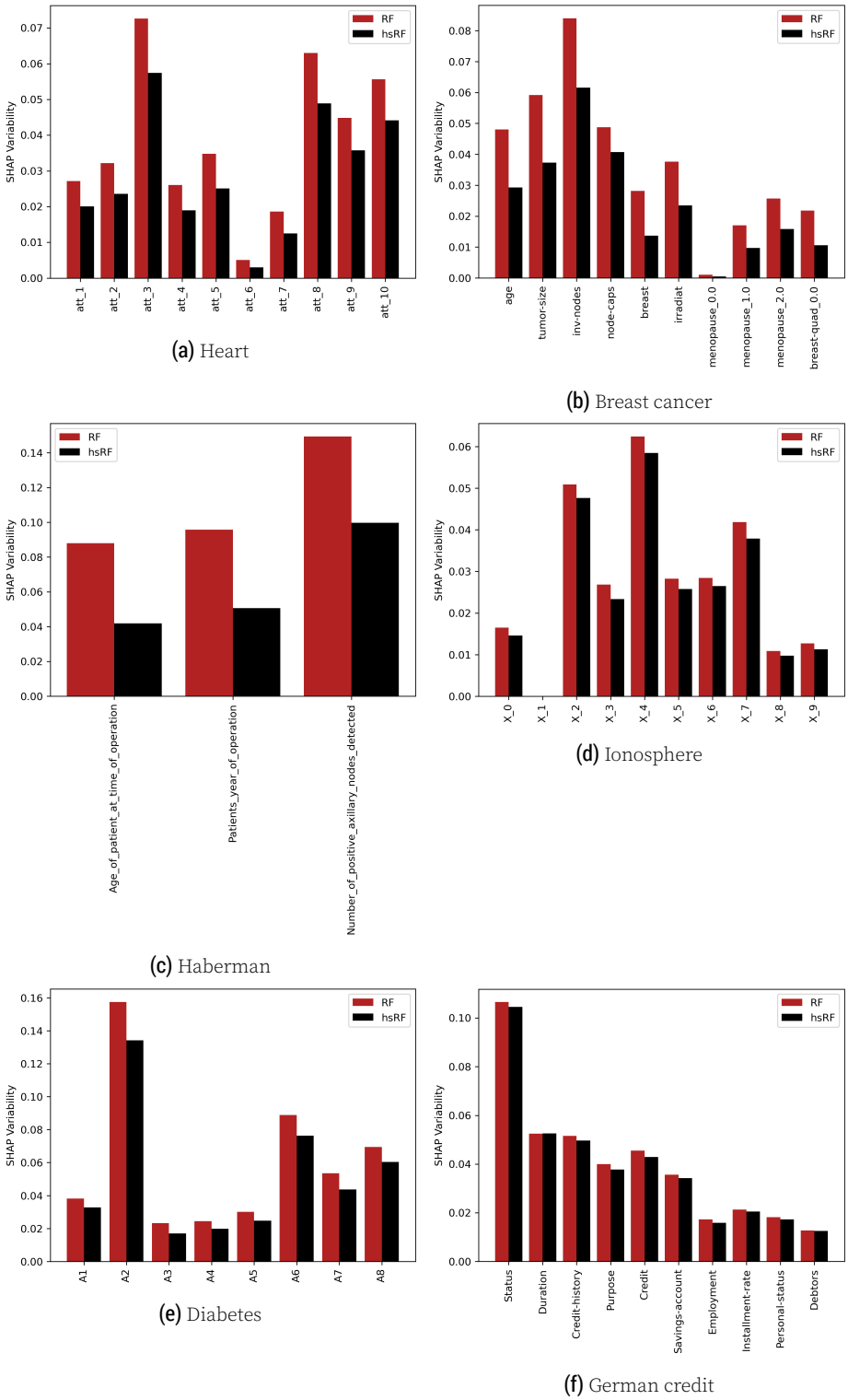
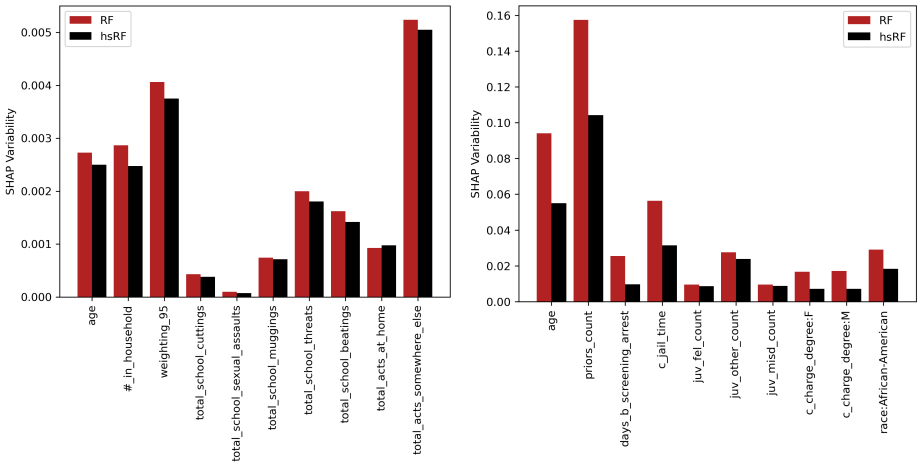


Figure 15. Comparison of SHAP value variabilities for the first six classification data sets before and after applying HS.



(a) Juvenile (b) Recidivism

Figure 16. Comparison of SHAP value variabilities for the last two classification data sets before and after applying HS.