Compressed Rule Ensemble Learning

Malte Nalenz
LMU Munich

Thomas Augustin
LMU Munich

Abstract

Ensembles of decision rules extracted from tree ensembles, like RuleFit, promise a good trade-off between predictive performance and model simplicity. However, they are affected by competing interests: While a sufficiently large number of binary, non-smooth rules is necessary to fit smooth, well generalizing decision boundaries, a too high number of rules in the ensemble severely jeopardizes interpretability. As a way out of this dilemma, we propose to take an extra step in the rule extraction step and compress clusters of similar rules into *ensemble rules*. The outputs of the individual rules in each cluster are pooled to produce a single soft output, reflecting the original ensemble's marginal smoothing behaviour. The final model, that we call *Compressed Rule Ensemble* (CRE), fits a linear combination of ensemble rules. We empirically show that CRE is both sparse and accurate on various datasets, carrying over the ensemble behaviour while remaining interpretable. Predictions can be explained by looking at the active ensemble rules, allowing external validation. We showcase that ensemble rules are also useful for a wider range of models that utilize decision rules extracted from tree ensembles.

1 INTRODUCTION

Ensemble methods that use decision trees as base learners are among the most popular and successful general-purpose supervised learning methods. They can naturally adapt to non-linearities, capture interactions between features and often perform well off-the-shelf with little to no parameter tuning (Fernandez-Delgado et al., 2014). Most tree ensemble methods use re-sampling schemes in order to create trees that capture different aspects of the training data. This increased model variance in return leads to more stable, robust and accurate predictions compared to a single decision tree.

However, the increase in model complexity resulting from the ensemble approach is also a major downside. While a single decision tree is straightforward to interpret, a forest resulting from the combination of hundreds, deep and randomized, decision trees can not be processed by the human mind, essentially turning the ensemble into a black-box model. Methods for analyzing the behaviour of the forest exist, such as Variable Importance (Breiman, 2001) and recently proposed variants, such as SHAP-values (Lundberg and Lee, 2017), but the intuitive structure of the individual trees is lost. This makes it unclear how exactly a decision is reached, which is a fact that is often not acceptable in high-stake situations, such as a medical treatment choice.

One approach towards interpretable machine learning models is to learn rule ensembles. As decision rules are composed of simple if-else statements, they are easier to interpret for humans compared with deep decision trees. One such approach is RuleFit introduced by (Friedman and Popescu, 2008). Instead of learning decision rules directly, the candidate rules are extracted from decision forests and combined in a penalized linear model. The rationale of RuleFit is both simple and compelling: Tree ensemble methods often have remarkable accuracy. However, their greedy learning procedure produces overly complicated models. By regularizing away the unnecessary complexity, RuleFit takes a step towards a favourable accuracy-complexity trade-off.

We argue that rule ensemble approaches suffer from competing interests: In order to provide smooth decision boundaries, a property essential for good generalization in ensembles (Bühlmann and Yu,

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

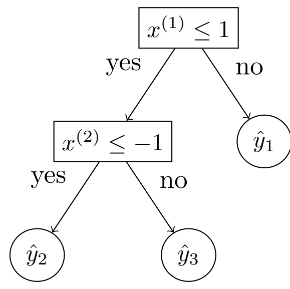
2002; Bühlmann, 2012), a sufficiently large number of slightly different – potentially overlapping – rules need to be selected for the final ensemble, which in return harms the interpretability. We propose a way to solve this dilemma based on the interpretation of ensemble learning as a smoothing of the hard thresholding behaviour of decision rules (Bühlmann and Yu, 2002). Instead of using the individual decision rules directly, we first identify clusters of similar conditions. To this end, univariate clustering is performed on the splitpoints in each covariate. The resulting groups of similar conditions are then compressed into soft conditions, which we call *ensemble conditions*. By averaging the discrete outputs of the individual conditions, ensemble conditions produce a smoother output, which reflects the behaviour of the original forest method. Ensemble rule compression allows to carry over the smoothing behaviour of forest methods while sacrificing very little in terms of interpretability and at the same time reflecting the uncertainty about the 'true' splitpoint. We argue that often already a few compressed rules allow us to capture and interpret the central behaviour of the forest, providing a glimpse into the black box.

The structure of this paper is as follows. In section 2 we give an overview of existing rule ensemble approaches, and in section 3 we review the RuleFit approach and introduce notations. Section 4 introduces *compressed rule ensembles (CRE)*, that combine ensemble rules, based on ensemble conditions, with the RuleFit approach. We also showcase that ensemble rules are useful in other rule ensemble frameworks. Section 5 presents our experiments on classification tasks. Section 6 concludes.

2 RELATED WORK

Several ways have been proposed to (greedily) induce decision rule ensembles. Classical approaches include the divide and conquer algorithms, which sequentially induces non-overlapping rules (Cohen, 1995; Fürnkranz, 1999), and boosted decision rules (Freund and Schapire, 1996; Weiss and Indurkhya, 2000; Dembczyński et al., 2008) that use re-weighting schemes to induce rules that iteratively reduce the error from the current ensemble.

RuleFit (Friedman and Popescu, 2008) combines candidate rules in a penalized linear model. This two-step formulation of rule learning allows the application of standard statistical learning methods. In its original formulation, rules are extracted from gradient boosted decision trees (Friedman, 2002), but also other forest types have been explored (Nalenz



$\phi(c)$	s	v	t
$I(x^{(1)} \leq 1)$	1	1	1
$I(x^{(1)} > 1)$	2	1	1
$I(x^{(1)} \leq 1)$	3	1	1
$I(x^{(2)} \leq -1)$	3	2	-1
$I(x^{(1)} \leq 1)$	4	1	1
$I(x^{(2)} > -1)$	4	2	-1

Figure 1: Left: Binary decision tree with 3 leafs, 1 internal node and the root node. Right: Further decomposition of the decision rules into the elementary conditions. Multiple conditions per rule are combined with the logical AND.

and Villani, 2018; Fokkema, 2020). Inducing decision rules jointly with learning the weight coefficients was explored in (Jawanpuria et al., 2011) and (Wei et al., 2019). Using quadratic programming to select the final ruleset was explored in (Meinshausen, 2010).

Another interesting way to combine decision rules was recently proposed with SIRUS (Bénard et al., 2021), where paths are extracted from an adapted version of random forests. There the data is quantile transformed beforehand to limit the possible splitpoints in trees, allowing to identify frequent patterns across trees. The most common decision rules are simply averaged to produce a prediction, without the need of a linear weighting and selection step, which significantly improves the model stability.

3 PREDICTIVE RULE ENSEMBLES

Given the N training examples $(y_i, x_i), i = 1, \ldots, N$, with generic variables y and x, where y is either discrete or numeric and $x = (x_1, ..., x_p) \in \mathbb{R}^p$ is the p-dimensional covariate vector, with the j'th component of x denoted as $x^{(j)}$, we seek to find a function, that allows to predict y from x. In the context of predictive rule ensembles and assuming a regression task we look at the class of generalized additive models

$$y = \sum_{h=1}^{H} \alpha_h r_h(x), \quad r_h \in \{0, 1\}, \tag{1}$$

where decision rules $r_h(x)$ are used as basis functions, weighted by the coefficients α_h. Instead of learning decision rules directly from the data, the RuleFit framework takes a two-step procedure.

First, a tree ensemble is generated. (Friedman and Popescu, 2008) use gradient boosting to generate the set of trees. As boosting shows great accuracy in many tasks, it is reasonable to assume that the model is able to find interesting subspaces, defined by decision rules. The decision trees are then decomposed into their defining decision rules and harvested across the whole ensemble. In our approach we decompose the rules further into their elementary conditions. s denotes the index of the rule that the condition originates from, v the index of the covariate used for the comparison and t the splitpoint. Figure 1 shows an example of this decomposition and the introduced notation. A condition is thus defined by the triplet $c_a = (s_a, v_a, t_a), a = 1, ..., A$, where A is the total number of conditions collected from the forest and $M = |\{s\}|$ is the number of decision rules. Note that in this step only the different paths to all nodes are stored, not the values of the leaf nodes. The full rule is the conjunction of its individual conditions. The hard-thresholding split function ϕ is given by

$$\phi(x, v, t) = I(x^{(v)} < t) \quad \text{or} \tag{2}$$

$$\phi(x, v, t) = I(x^{(v)} \geq t) = 1 - I(x^{(v)} < t) \tag{3}$$

depending on the direction that is encoded in r_h and assuming numerical features. As the second step the decision rules are included, together with linear terms, as 0-1 features in a linear regression model. Using the above notations, the full rules $r_h \in \{0, 1\}$ are obtained by taking the product of the conditions that are part of rule h,

$$r_h(x) = \prod_{a:s_a=h} \phi(x, v_a, t_a). \tag{4}$$

The original forest contains a large number of rules. L1-regression (Tibshirani, 1996) is used to reduce the large set of candidate rules to the truly predictive ones.

4 COMPRESSED RULE ENSEMBLES (CRE)

As in RuleFit, in a first step a tree ensemble is generated. Either the random forest or gradient boosting framework can be applied. For computational efficiency we use XGBoost (Chen and Guestrin, 2016) to generate the rules. However, before transforming the rules directly into 0-1 features using (4), we take an additional step and compress groups of similar conditions into ensemble conditions.

4.1 Ensemble Compression

When using re-sampling techniques commonly featured in random forests and (stochastic) gradient boosting, the splitpoints inside the forests will often appear in clusters. Depending on the sample that is seen by a tree and the weights in this iteration, the tree induction algorithm will often lead to similar trees with slightly different splitpoints. This is generally beneficial in terms of predictive performance, as it leads to a smooth decision boundary (Bühlmann and Yu, 2002), which stabilizes predictions. However, this also implies that when removing many rules in RuleFit we expect the decision boundaries to become non-smooth and the predictive performance to drop. Therefore, the goal is to preserve smoothness but reshape it in a form that is accessible for human interpretation.

4.2 Clustering of Similar Conditions

To preserve the forest behaviour, we identify clusters of similar conditions that only differ in their exact splitpoint and combine their binary decision into a single smooth decision. More formally, for each covariate $j, j = 1, ..., p$ we look at the vector of splitpoints $T^{(j)} = (t_a : v_a = j)$. This step collects all splitpoints from splits involving covariate j from all rules that were extracted from the original forest. Note that the splitpoints are taken from single condition rules or from more complicated rules, involving several conditions and other covariates. Also, no attention is drawn to the depth of the rule in which the condition appears, as with the symmetry in the conjunctive form of decision rules, ordering is somewhat arbitrary. As we expect the clusters of splitpoints to be fairly obvious, we use k-means as a robust and well-understood clustering method to find the clusters. We assume that the splitpoints will appear in a relatively small number of clusters. The k centers in the k-means algorithm are chosen to minimize the intra-cluster variation,

$$C(k, \mu, T^{(j)}) = \sum_{l=1}^{k} \sum_{\{z:g_z^{(j)}=l, t_z \in T^{(j)}\}} (t_z - \theta_l)^2 \tag{5}$$

where $z \in \{1, ..., Z = |T^{(j)}|\}$ is the index of splitpoints for covariate j, $g^{(j)} = (g_1, ..., g_Z)$ is the vector of cluster labels for the splitpoints and $\theta = (\theta_1, ..., \theta_k)$ the vector of mean values of the k groups. For this one-dimensional clustering problem, the *Ckmeans.1d.dp* algorithm (Wang and Song, 2011) can be applied, which uses dynamic programming to find the global optimal solution. If a certain splitpoint is very important in the prediction task, it will often appear in the vector $T^{(j)}$ and dominate the cluster solution in equation (5). This is a desired property, resulting in an implicit weighting of regions found important by the forest method. As the appropriate number of clusters for each covariate is unknown apriori, we determine the optimal k using the *AIC* criterion with a pre-specified maximum number of clusters k_{max}. Other clustering algorithms that we considered were Gaussian-Mixture-Models and density-based clustering methods, such as DBSCAN (Schubert et al., 2017). As the results were quite similar, we decided to stick with k-means as the most simple and robust approach. Note that the clustering is performed on the splitpoints found in the forest, never on the original data, making this step computationally cheap. For 500 trees, the number of splits is typically << 1000 per covariate and therefore almost independent from N and only linear in p.

4.3 Combining Multiple Conditions into a Soft Condition

Given the vectors of splitpoints for group l of covariate j from the clustering step, $T_l^{(j)} = (t_i \in T^{(j)} : g_i^{(j)} = l)$, we combine the individual conditions to ensemble conditions. The combined output of the ensemble condition is computed by average pooling of the outputs from the individual conditions. The soft output function for ensemble condition l becomes

$$\Phi(x, v, l) = |T_l^{(v)}|^{-1} \sum_{t \in T_l^{(v)}} \phi(x, v, t). \qquad (6)$$

Averaging over several conditions turns the individual binary outputs $\phi(x, v, t) \in \{0, 1\}$ into a soft output $\Phi(x, v, l) \in [0, 1]$. In contrast to other soft decision rule approaches, such as Akdemir et al. (2013), our approach is non-parametric. Φ reflects the empirical distribution from the splits found in the forest and preserves the univariate behaviour of the full ensemble, and compresses it in a single ensemble condition. Figure 2 shows the distribution of splitpoints for the Diabetes dataset from the UCI repository (Dua and Graff, 2017) and the clustering result using $k_{max} = 4$. We observe that the $\Phi(x)$ can follow arbitrary distributions that are quite different from logistic curves. Also, note the dense regions, forming clusters of splitpoints that are interesting for predictions. Lastly, suppose the underlying relationship is in fact a stepfunction. In that case, we expect the forest method to capture it, and in return, the clusters' intervals become very narrow.

To finish this step, all original conditions are replaced by their corresponding ensemble conditions. Using ensemble conditions turns each binary rule r_h into a smooth rule $\mathcal{R}_h$ generalizing equation (4). The output of $\mathcal{R}_h$ is calculated via

$$\mathcal{R}_h(x) = \prod_{a: s_a = h} \Phi(x, v_a, g_a) \in [0, 1]. \qquad (7)$$

As all conditions in each cluster have the same output for any given x_i, ensemble compression allows to safely remove a large number of redundant rules.

4.4 Finding a Sparse Set

Given the ensemble rules, the second step combines them to a reduced ensemble. We investigate two ways of rule aggregation, weighting and averaging.

4.4.1 Linear Weighting

Following RuleFit, the ensemble rules are included, together with linear terms, in the (generalized) linear regression model:

$$F(x) = \sigma(\beta_0 + \sum_{j=1}^{p} \beta_j x_j + \sum_{h=1}^{H} \alpha_h \mathcal{R}_h(x)), \qquad (8)$$

where σ is a link function. As in (Friedman and Popescu, 2008) the rule terms are not scaled, leading to a higher penalty on rules with low support. However one property specific to compressed rules is that rule support decreases slower with additional conditions, leading to lower penalization of complicated rules. As complicated rules are highly undesireable in terms of interpretability, we counteract this effect by decreasing the scale of each $\mathcal{R}_h$ proportional to the number of conditions involved, via

$$\mathcal{R}_h^*(x) = \frac{\mathcal{R}_h(x)}{length(\mathcal{R}_h)^{\eta}}, \quad \eta > 0, \qquad (9)$$

where η is a parameter that controls the amount of extra penalty for the number of conditions involved and $length(\mathcal{R}_h)$ is the number of conditions. This is similar to the rule structured prior used in (Nalenz and Villani, 2018). Penalizing depth was also found an effective way to promote simplicity in (Wei et al., 2019; Chipman et al., 2010). We found $\eta = 0.5$ to work well as a default choice, but η can also be guided by prior knowledge, about the complexity of the underlying relationship or tuned via cross validation. The weights are found by solving the L1-regularized regression

$$\{\alpha^*, \beta^*, \beta_0^*\} = \underset{\beta_0, \beta, \alpha}{\arg\min} \Bigg[L(y, F(x)) + \qquad (10)$$

$$\lambda \left(\sum_{j=1}^{p} |\beta_j| + \sum_{h=1}^{H} |\alpha_h| \right) \Bigg], \qquad (11)$$

with L being an appropriate loss function. A big advantage of the linear model approach is its easy interpretability. Following (Friedman and Popescu, 2008), we can rank rules and linear terms by their (rescaled) effect size $|\alpha^*|$ and $|\beta^*|$ respectively as a measure of importance. In our experiments we use the R-package (R Core Team, 2021) *glmnet* (Friedman et al., 2010), and the penalty parameter λ is chosen via cross-validation (CV). A popular choice is to use λ_{1se} the highest λ value within one standard deviation of the minimum, in order to promote sparsity, which is also used as a standrad choice for CRE.

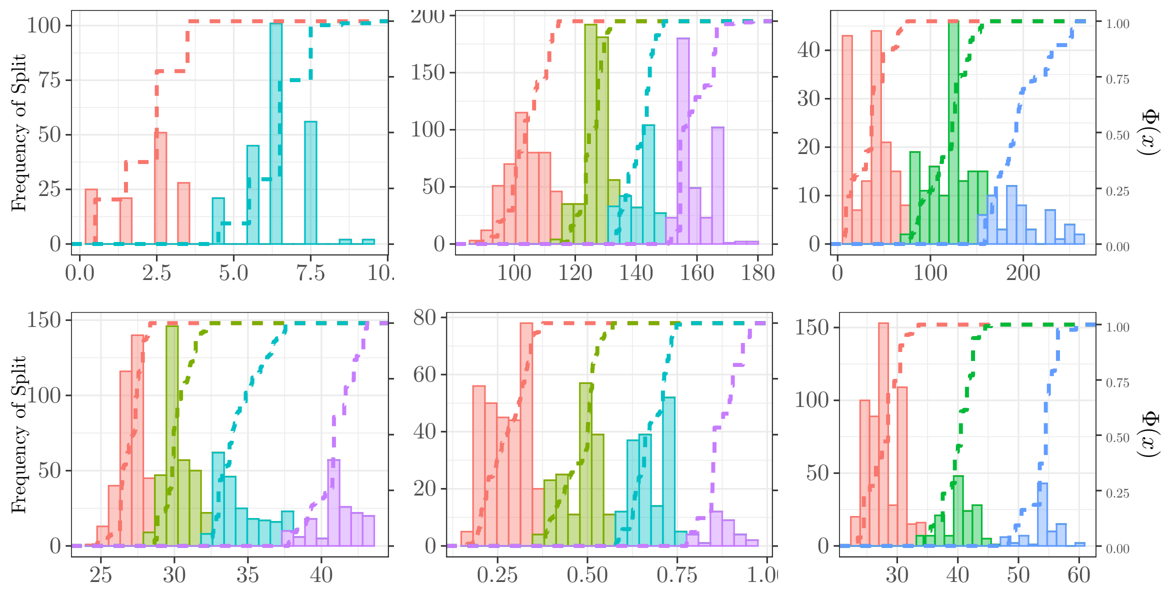


Figure 2: Distribution of splitpoints for the 6 most used covariates in the diabetes data set (cf. Section 5.3. and the supplementary material), using $k_{max} = 4$. Colours indicate the cluster solution from the *Ckmeans.1d.dp* algorithm. The dashed line shows the soft output $\Phi(x)$ for each cluster (compressed condition).

4.4.2 Averaging

An alternative to the linear combination (8) is to simply count the number of occurrences of each smooth rule $\mathcal{R}_h$ and average over the most frequent rules. For each rule the associated prediction values for cases that are covered/not covered by a rule, μ_+, μ_- respectively are the weighted mean on the training data

$$\mu_{+,h} = \frac{1}{\sum_{i=1}^{N} \mathcal{R}_h(x_i)} \sum_{i=1}^{N} y_i \mathcal{R}_h(x_i), \quad (12)$$

$$\mu_{-,h} = \frac{1}{\sum_{i=1}^{N} (1 - \mathcal{R}_h(x_i))} \sum_{i=1}^{N} y_i (1 - \mathcal{R}_h(x_i)), \quad (13)$$

assuming $y \in \{0, 1\}$, which is a soft version of the SIRUS algorithm. Predictions for the whole ensemble rule are obtained via

$$\hat{y}_{i,h} = \mathcal{R}_h(x_i, v, g) \cdot \mu_{+,h} + (1 - \mathcal{R}_h(x_i, v, g)) \cdot \mu_{-,h}. \quad (14)$$

The output of the whole ensemble is then simply the average of the K most frequent compressed rules $\mathcal{R}_h$. Adopting the SIRUS approach (Bénard et al., 2021) to use compressed conditions instead of crisp decision rules goes together quite naturally with the idea of ensemble compression, avoiding any data discretization. We find the core idea of SIRUS particularly interesting, as it can be seen as a proxy of how good a small number of ensemble rules can summarise a whole tree ensemble.

4.5 The Effect of Ensemble Compression

4.5.1 Choice of k_{max}

The inverse k_{max}^{-1} can be interpreted as compression rate. Setting $k_{max} = 1$ compresses all splitpoints per covariate into a single group and results in a monotonic transformation of the covariate, based on the distribution of the splitpoints. In this setting, only monotonic effects can be captured and no change of sign is possible. Increasing k_{max} allows changes in sign and magnitudes of the effects, therefore finding different regions of interest. As $k \to Z$, where Z is the number of splitpoints in this covariate, our model approaches the original RuleFit model. Using ensemble compression also acts as a regularizer, as it makes it harder to overfit on individual rules but has to take into account the general pattern found by the forest. The choice of k_{max} can be guided by visual inspection using histogram plots as in Figure 2, or by considering the desired trade-off. For a higher degree of interpretability, a low k_{max} should be used. Another option is to rely on cross-validation and examine the different choices in terms of their expected accuracy-complexity trade-

Table 1: Runtime in s. (Data is taken from the OpenML repository (Vanschoren et al., 2013)).

Dataset	N	CRE_{k2}	CRE_{k4}	RuleFit
Magic	19022	174	230	2633
EEG	14980	512	740	2243

off. We empirically found a relatively small value of k_{max} (e.g. $k_{max} \in \{2, 3, 4\}$) is usually a good choice, whereas the gain in accuracy by choosing a higher k is typically incremental and out-weighted by the increase in model-complexity (cf. Section 5).

4.5.2 Computational Cost

The number of unique conditions is reduced to a maximum of k_{max} distinct ensemble conditions for each covariate, which can be significantly lower than the number of distinct original conditions. This also leads to a much smaller number of unique rules in the linear modelling step, which is important, as the design matrix in equation (11) is of size $(n, p+H)$. As duplicates and colinear terms can be safely removed, this effectively lowers the computational cost and memory usage significantly, and decreasing k_{max} lowers the computation time considerably. Table 1 shows the runtime on two medium-sized datasets, comparing RuleFit and CRE using $k \in \{2, 4\}$. On these datasets, the speed-up compared to RuleFit is quite significant. Interestingly, on the Magic dataset, the speedup is much more notable as the number of predictive covariates is relatively small and around 60 % of $\mathcal{R}_h$ can be removed from the initial set (using $k_{max} = 4$). If the splits distribute evenly over many covariates, as in the EEG data, the reduction is smaller but still notable.

4.5.3 Interpretation of Ensemble Conditions

Ensemble conditions consist of a typically large number of thresholds, and interpreting the exact split points would be a tedious task. For a literal description we suggest to instead look at summary statistics of T, such as the mean, $\zeta = \bar{T}$, or $\zeta = \text{median}(T)$. The rule can then be read as $x \lessapprox \zeta$ (*less than around* ζ), instead of the precise $x < s$ counterpart. Another possibility is to provide the interval $[\min(T), \max(T)]$. For a more detailed insight in the distribution of interesting rules, one can instead use histograms such as the one shown in Figure 2. We argue that providing an exact split point in the finite data domain neglects the uncertainty involved about what other choices would be reasonable. Compressed rules offer a natural way to communicate the model uncertainty. If the original ensemble shows a large spread of splitpoints, this is reflected in larger intervals, whereas in the case of clear cut-points, the intervals will become narrow.

5 RESULTS

In this section, we test our method empirically. The goal is to show that CRE is able to produce both accurate and small models due to the smooth boundaries introduced by the ensemble compression.

5.1 Experimental Setup

For comparison, we use 16 binary classification datasets from the UCI repository (Dua and Graff, 2017). We chose datasets that consist of mostly numerical covariates and require minimal preprocessing. A detailed description of selection criteria and preprocessing, algorithm settings and additional results can be found in the supplementary material (SM). We limit the experiments to binary classification but note that CRE can also be extended to regression, multi-label and multi-target classification (Aho et al., 2012).

5.2 Competing Methods

As a black box baseline with generally strong predictive performance we include random forests and gradient boosting. Random forest (RF) is run with default settings using the original `randomForest` R-package (Breiman, 2001). Gradient boosting, implemented with the `xgboost` R-package (Chen and Guestrin, 2016), is more dependent on parameter tuning. We use model based optimization with `mlrMBO` (Bischl et al., 2018) inside each fold, in order to find reasonable parameters and ensure a fair comparison.

We compare against two versions of RuleFit, both implemented with the `pre` R-package (Fokkema, 2020). RuleFit uses normal CART trees as base learners and parameter settings that most closely resemble the original version of RuleFit. In order to determine a reasonable tree depth, we use 5-fold CV inside each fold. PRE is a more interpretable setting proposed in (Fokkema, 2020) that uses λ_{1se} and an average tree depth of 3 (without tuning).

SIRUS is built using the `sirus` (Bénard et al., 2021) R-package, with the number of rules determined using the CV strategy proposed by the authors.

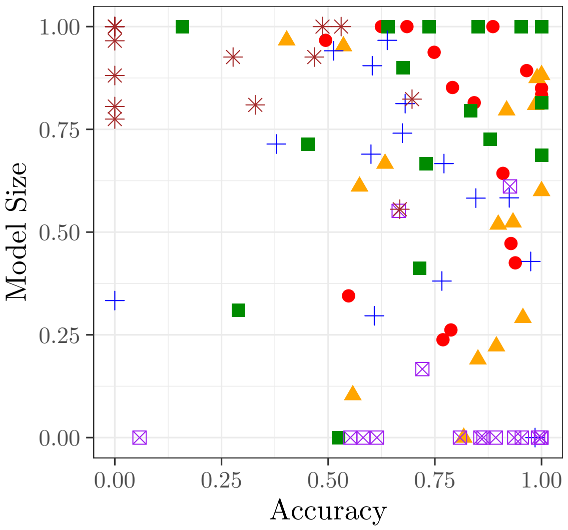
Also we include the recently proposed Generalized Rule Model GLR (Wei et al., 2019) in our comparison, using the `aix360` python library (Arya et al., 2019). The parameters were chosen following the cross-validation strategy proposed by the authors.

Table 2: Accuracy measured in AUC for the competing methods on the 16 benchmark datasets.

Data	CRE_S	$CRE_{k:2}$	$CRE_{k:4}$	$CRE_{k:6}$	CRE_{RF}	GLR	PRE	RF	RuleFit	SIRUS	XGB
(1) Australian	0.901	0.937	0.939	0.944	0.935	0.942	0.930	0.936	0.938	0.921	0.939
(2) Banknote	0.986	1	1	1	1	1	1.000	1.000	1.000	0.972	1
(3) Biodeg	0.871	0.929	0.931	0.930	0.919	0.926	0.915	0.938	0.924	0.840	0.932
(4) Blood Transf	0.732	0.730	0.750	0.748	0.741	0.751	0.724	0.668	0.751	0.708	0.746
(5) Diabetes	0.823	0.830	0.830	0.831	0.832	0.844	0.829	0.825	0.840	0.807	0.841
(6) Haberman	0.712	0.683	0.670	0.680	0.621	0.662	0.676	0.685	0.708	0.651	0.688
(7) Heart	0.898	0.910	0.896	0.896	0.888	0.898	0.877	0.905	0.897	0.899	0.906
(8) ILPD	0.709	0.728	0.723	0.718	0.704	0.736	0.735	0.752	0.706	0.729	0.724
(9) Ionosphere	0.955	0.963	0.964	0.965	0.954	0.948	0.965	0.981	0.968	0.941	0.970
(10) Liver	0.649	0.666	0.679	0.644	0.667	0.646	0.623	0.564	0.657	0.644	0.654
(11) Parkinsons	0.906	0.945	0.959	0.950	0.968	0.915	0.857	0.953	0.960	0.888	0.962
(12) Pop Failure	0.907	0.947	0.945	0.952	0.946	0.942	0.947	0.920	0.925	0.889	0.946
(13) Sonar	0.863	0.923	0.927	0.910	0.925	0.864	0.875	0.949	0.915	0.829	0.940
(14) Spambase	0.963	0.985	0.986	0.986	0.985	0.979	0.980	0.987	0.985	0.933	0.988
(15) WBCD	0.991	0.992	0.993	0.991	0.993	0.991	0.992	0.992	0.989	0.981	0.995
(16) Wilt	0.952	0.990	0.992	0.991	0.993	0.969	0.991	0.990	0.993	0.901	0.990
$\bar{r}$	8.500	5.156	4.469	5.156	5.656	5.969	7.562	5.250	5.375	9.688	3.219
ΔAUC	0.034	0.012	0.011	0.014	0.018	0.021	0.028	0.019	0.012	0.051	0.008

Table 3: Left: Number of coefficients selected for the final model. Right: Figure 3: Normalized Accuracy vs. Normalized Sparsity where 1 is the best on each dataset and 0 the worst. $CRE_{k:2}$ (red circle), $CRE_{k:4}$ (orange triangle), GLR (green squares), PRE (blue cross), RuleFit (purple squares), SIRUS (brown stars).

Data	$CRE_{k:2}$	$CRE_{k:4}$	$CRE_{k:6}$	CRE_{RF}	GLR	PRE	RuleFit	SIRUS
(1)	18	26	32	19	13	20	40	15
(2)	11	21	29	18	5	45	45	14
(3)	52	62	69	51	45	50	106	22
(4)	7	8	8	6	11	9	22	6
(5)	9	18	23	10	14	28	36	9
(6)	2	3	3	2	8	4	23	6
(7)	13	17	18	18	10	22	28	18
(8)	10	10	15	9	14	10	68	8
(9)	33	40	43	22	14	23	27	15
(10)	7	9	10	8	17	8	24	10
(11)	22	25	27	20	68	14	35	18
(12)	29	38	40	27	10	25	46	17
(13)	50	61	61	42	48	31	54	19
(14)	95	112	119	121	48	75	149	22
(15)	31	32	36	24	22	28	36	15
(16)	16	23	29	23	7	55	91	17
$\bar{r}$	3.38	5.13	6.22	3.44	3.44	4.44	7.56	2.41



The CRE based models use gradient boosted trees from `xgboost` to generate the rules. Different degrees of compression are tested, using $k_{max} = \{2, 4, 6\}$ denoted as $CRE_{k:k_{max}}$. Only CRE_{RF} uses random forests to generate the rules, as a way to measure the influence of the tree generating process, and $k_{max} = 4$. All CRE models use $\eta = 0.5$ (cf. (9)) to promote taking in less complex rules. No parameters, for the rule generation ,η or k_{max}, are tuned. Better predictive performance may be reached, but we are interested in the 'out-of-the-box' performance in this article. We also test compressed rules with averaging of the rules, which resembles the SIRUS approach. To estimate the influence of the rule compression, CRE_S uses the average number of rules used by SIRUS on each dataset, leading to the overall same model complexity as SIRUS. We expect the same number of ensemble rules to generalize better compared to binary rules. An open-source R-implementation of the CRE methods used in this article is available under `https://github.com/maltenlz/Compressed-Rule-Ensembles`.

5.2.1 Accuracy

We report accuracy as measured by the area under the curve (AUC). Table 2 shows the results over the 16 datasets, together with the mean rank $\bar{r}$ and average deviation from the best AUC value (ΔAUC) over all datasets. Using the multiple comparison approach from Demšar (2006); Calvo and Santafé Rodrigo (2016), the post-hoc Friedman-test for overall differences in AUC indicates significant difference between algorithm performance ($p = 1.9e^{-7}$). A Nemenyi-test indicates no statistically significant difference in performance between XGB, all CRE versions, RF, RuleFit and GLR ($\alpha = .05$). The same is not true for PRE, CRE_S and SIRUS, which perform significantly worse. In line with our expectation and previously reported results, a well-tuned XGB model is on average the most accurate. $CRE_{k:4}$ achieves the second-best rank, outperforming all rule-based competitors, the vanilla random forest and the tuned RuleFit model.

$CRE_{k:2}$ is very competitive in terms of accuracy to XGB, RF and RuleFit and performs similar or better than the other rule-based competitors, despite using a high degree of compression (using only 2 clusters per covariate). Using a higher compression parameter $CRE_{k:6}$ is not beneficial in the analyzed datasets. This can be attributed to the regularizing effect of ensemble compression, making $k_{max} = 4$ our recommended default choice for prediction while in most data situations already $k_{max} = 2$ suffices.

As seen by the average deviation from the best method, CRE and RuleFit are on average not much behind the best method, implying a stable performance. GLR also shows good overall performance but has some datasets where the difference to the best method is quite large. SIRUS, CRE_S and PRE sacrifice on average a notable amount of accuracy but still are competitive on some datasets (e.g. WBCD and ILPD).

5.2.2 Model Complexity

Table 3 shows the number of selected rules and linear terms. In terms of model complexity, SIRUS, $CRE_{k:2}$, CRE_{RF}, GLR and PRE produce the smallest models, with SIRUS being the winner (note that CRE_S uses the same number of rules). However, as discussed above, SIRUS and PRE sacrifice a substantial amount of accuracy to achieve this goal, whereas $CRE_{k:2}$ on most datasets produces similarly sparse models while remaining competitive concerning predictive performance. $CRE_{k:4}$ takes in slightly more rules but also produces reasonably small models on most datasets, while maintaining strong accuracy. RuleFit produces overall the largest models, often by quite a large margin.

5.2.3 Accuracy vs. Sparsity

The trade-off between accuracy and model size can be seen in Figure 3, where only $CRE_{k:2}$, $CRE_{k:4}$ and GLR are able to consistently achieve good accuracy and sparsity jointly. We conclude that CRE produces both accurate and sparse models, whereas most of the competing methods have to compromise either aspect. GLR and PRE also show overall very reasonable trade-offs under the competing methods. If one is willing to sacrifice some accuracy, also SIRUS and CRE_S become very reasonable choices as highly interpretable but slightly less accurate models. The standard RuleFit shows good accuracy but leads to clearly sub-optimal trade-offs..

To summarise, while in this study a well-tuned XGB model is the most accurate, CRE is on average not far behind while producing small model sizes. The good performance of CRE is enabled through the usage of ensemble rules allowing smooth decision boundaries even for sparse solutions. Ensemble compression also improves the predictive performance of the SIRUS framework when using the same amount of rules. CRE produces more accurate models in terms of AUC when combined with gradient boosting, while producing more sparse solutions when using random forest to generate the rules.

However, we also found that ensemble compression is not always effective. For example, on the Ionosphere dataset CRE performs worse than RuleFit both in terms of accuracy and model size. We found this to happen on some physical datasets when the underlying relationship might not be smooth. In this case, CRE does not improve performance. It is reassuring that even in these suboptimal data situations CRE still remains competitive.

5.3 Interpretation

Finally, we showcase how CRE allows for vivid interpretation. Here we focus on the literal interpretation of the rules, as they are the main advantage of rule ensembles. Table 4 shows the output of $CRE_{k:4}$ for the Diabetes dataset [1] using $[\min(T), \max(T)]$ as summary (cf. 4.5.3). Shown here are the rescaled coefficients $\hat{\beta}$ as a measure of impact when the rule 'fires'. An alternative would be to use $N^{-1}\Phi(x)\hat{\beta}$ as a mea-

[1] A description of the covariates is available in the supplementary material. Y = 1: diabetes positive.

Table 4: Model output for the Diabetes data.

Rule	$\hat{\beta}$
Intercept	−1.41
age ≥ [21.5;26.5] ∧ BMI ≥ [21.75;28.15]	0.57
age ≥ [27.5;34.5]	0.47
BMI ≥ [21.75;28.15]	0.38
BMI <[28.45;32.35] ∧ preg <[5.5;6.5]	−0.26
BMI <[38.45;45.45] ∧ pedi <[0.6;0.9]	−0.26
BMI ≥ [21.75;28.15] ∧ pedi ≥ [0.14;0.37]	0.22
linear: plas	0.15
plas <[115.5;141.5] ∧ preg <[5.5;6.5]	−0.09
BMI <[38.45;45.45] ∧ plas <[142;188.5]	−0.06
BMI ≥ [28.45;32.35] ∧ pedi ≥ [0.38;0.59]	0.06
preg ≥ [5.5;6.5]	0.05
BMI <[38.45;45.45] ∧ plas <[142;188.5] ∧ preg <[7.5;8.5]	−0.05
BMI <[28.45;32.35] ∧ preg <[7.5;8.5]	−0.05
preg <[7.5;8.5] ∧ skin ≥ [3.5;11.5]	−0.02

sure of global importance. The rules show that diabetes is strongly connected to age and BMI and its interaction. BMI appears to be the most important covariate, appearing in almost all ensemble rules, often in interaction with different covariates. The rule $BMI \geq [21.75; 28.15]$ also demonstrates the usefulness of ensemble rules. In this region, the risk of diabetes starts to increase, and no single split value would describe the relationship well. The distribution of split points and $\Phi(x)$ can be visualized (cf. Figure 2).

Table 5: CRE prediction explanation.

Rule	$\hat{\beta}$	$\mathcal{R}(x)$	$\hat{\beta}\mathcal{R}(x)$
linear: plas	0.023	62.1	1.440
Intercept	−1.410	1	−1.410
age ≥ [27.5;34.5]	0.470	0.450	0.210
BMI <[38.45;45.45] ∧ pedi <[0.6;0.9]	−0.260	0.410	−0.110
preg ≥ [5.5;6.5]	0.050	1	0.050
BMI <[38.45;45.45] ∧ plas <[142;188.5]	−0.060	0.200	−0.010
BMI ≥ [21.75;28.15]	0.380	0.020	0.010
age ≥ [21.5;26.5] ∧ BMI ≥ [21.75;28.15]	0.570	0.020	0.010

CRE can also give an explanation of how a prediction is produced. Table 5 shows the output for a ‘close call’ observation with $(age, BMI, pedi, preg, plas) = (32, 23.3, 0.67, 8, 62.1)$. It is interesting to take a closer look at the rule $age \geq [27.5, 34.5]$. If this ensemble rule fully fires ($\mathcal{R}(x) = 1$, cf. equation (4)) the risk of diabetes increases by $\exp(0.47) = 1.6$. In this example, about half the splitpoints in the ensemble rule fire, leading to an increase in the risk of $\exp(0.21) = 1.24$. If hard rules were used instead, the rule could only give the full risk increase or none. While the last rule contributes little to the current prediction, it is still interesting: If the covariate BMI increases, this rule will fire more strongly, and the diabetes risk will increase. Instead of giving all or nothing decisions, CRE allows to spot grey areas that may be interesting for interventions.

6 CONCLUSION AND FUTURE WORK DIRECTIONS

We proposed a framework to compress decision tree ensembles into smooth decision rules. Combining ensemble conditions with the RuleFit approach leads to simpler and more robust models while being competitive in terms of predictive performance. As decision rules are a fundamental building block of many learning algorithms, rule compression could be beneficial for a wider range of models, as an easy way to provide smooth and stable decision boundaries.
We argue that the increase in complexity, due to smooth decision rules, does not harm interpretability. On the contrary, it more closely resembles human intuition and communicates the model's uncertainty about the exact rule.

We expect CRE to be more stable than RuleFit, as the ensemble rules depend less on the specific data sample and are more consistent between runs. However, in this paper, we were unable to test the stability empirically, as to the best of our knowledge, no suitable stability measure exists to measure literal similarity with continuous thresholds. The approach in (Bénard et al., 2021) requires discretizing the data, which does not make sense for CRE. Our preliminary experiments with stability metrics that try to measure the literal similarity between rulesets using continuous thresholds were unsatisfactory. Suitable stability measures along the lines of (Nair et al., 2021), but tailored for continuous thresholds, would be highly desirable as a direction for future work.

Another potential application could be to use rule compression to get an insight in the inner workings of a forest, by extracting the most common paths in the forest, as was showcased by the combination with the SIRUS approach. Future work will focus on the summary potential of ensemble compression as a tool for black-box interpretation by detecting and visualizing frequent rules across the forest, rather than building a new predictive model.

Acknowledgements

We thank the five anonymous reviewers and the editor for their constructive and helpful comments.

References

Aho, T., Ženko, B., Džzeroski, S., Elomaa, T., and Brodley, C. (2012). Multi-target regression with rule ensembles. *Journal of Machine Learning Research*, 13(8).

Akdemir, D., Heslot, N., and Jannink, J.-L. (2013). Soft rule ensembles for supervised learning. *stat*, 1050:22.

Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q. V., Luss, R., Mojsilović, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varshney, K. R., Wei, D., and Zhang, Y. (2019). One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv*.

Bénard, C., Biau, G., Da Veiga, S., and Scornet, E. (2021). Sirus: Stable and interpretable rule set for classification. *Electronic Journal of Statistics*, 15:427–505.

Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J., and Lang, M. (2018). mlrmbo: A modular framework for model-based optimization of expensive black-box functions. *arXiv*.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Bühlmann, P. (2012). Bagging, boosting and ensemble methods. In *Handbook of Computational Statistics*, pages 985–1022. Springer.

Bühlmann, P. and Yu, B. (2002). Analyzing bagging. *The Annals of Statistics*, 30:927–961.

Calvo, B. and Santafé Rodrigo, G. (2016). scmamp: Statistical comparison of multiple algorithms in multiple problems. *The R Journal, Vol. 8/1, Aug. 2016*.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4:266–298.

Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier.

Dembczyński, K., Kotłowski, W., and Słowiński, R. (2008). Maximum likelihood rule ensembles. In *International Conference on Machine Learning*, pages 224–231.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Fernandez-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15:3133–3181.

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software*, 92:1–30.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–22.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378.

Friedman, J. H. and Popescu, B. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2:916–954.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13:3–54.

Jawanpuria, P., Jagarlapudi, S. N., and Ramakrishnan, G. (2011). Efficient rule ensemble learning using hierarchical kernels. In *International Conference on Machine Learning*, pages 161–168.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774.

Meinshausen, N. (2010). Node harvest. *The Annals of Applied Statistics*, 4:2049–2072.

Nair, R., Mattetti, M., Daly, E., Wei, D., Alkan, O., and Zhang, Y. (2021). What changed? Interpretable model comparison. In *International Joint Conference on Artificial Intelligence*. IJCAI.

Nalenz, M. and Villani, M. (2018). Tree ensembles with rule structured horseshoe regularization. *The Annals of Applied Statistics*, 12:2379–2408.

R Core Team (2021). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Schubert, E., Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. (2017). DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, 42:1–21.

Smith, J. W., Everhart, J. E., Dickson, W., Knowler, W. C., and Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, pages 261–265. American Medical Informatics Association.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58:267–288.

Turney, P. D. (1994). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

Wang, H. and Song, M. (2011). Ckmeans.1d.dp: optimal k-means clustering in one dimension by dynamic programming. *The R Journal*, 3:29.

Wei, D., Dash, S., Gao, T., and Gunluk, O. (2019). Generalized linear rule models. In *International Conference on Machine Learning*, pages 6687–6696.

Weiss, S. M. and Indurkhya, N. (2000). Lightweight rule induction. In *International Conference on Machine Learning*, pages 1135–1142.

Supplementary Material: Compressed Rule Ensemble Learning

A Data Gathering and Preprocessing

Datasets, used in the section 5 and the diabetes dataset in section 4 of the paper are taken from the UCI machine learning repository (Dua and Graff, 2017). These criteria are:

- In this article we only consider binary classification.
- We chose datasets with mostly numerical features or features with low cardinality.
- Only datasets with low number of missing values are considered to minimize algorithm differences in missing value handling.

This criteria are set in order to make preprocessing as minimal as possible. Mostly numerical features are chosen for two reasons: (1) Ensemble compression only works on numerical features. (2) Tested algorithms have different ways to deal with discrete features, therefore we want to limit the influence of the implementation on the results. The preprocessing takes the following steps:

- Missing values are mean-imputed.
- Categorical features are simply transformed to numerical features, using the factor levels. (only in the Australian dataset).
- Dummy covariates are left as they are.
- For the liver dataset the Covariate ”drinks number” is used to generate the classes, as in (Turney, 1994).
- for the EEG-EYE data used in section 4.5.2 colinear covariates were removed due to numerical instabilities in the linear modeling step (for all methods).

Generally, first the datasets were selected and the preprocessing fixed, then we ran the experiments and no further datasets were excluded.

B Algorithm Settings

The exact settings and software used to allow reproducibility of results in section 4 are stated below:

- RuleFit: we use the R-package **pre** (Fokkema, 2020) to build the RuleFit model. For reasons of comparability we use boosted CART trees to generate the rules, but note that using the conditional random forest method to generate the trees might improve performance, as shown in Fokkema (2020). Other settings are set to the ones in Friedman and Popescu (2008). The most impactful parameter, *treedepth* is determined via internal 5-fold CV trying the values $1, 2, 3, 4, 5$. λ is taken as minimal value from the sequence, promoting accurate models.

- PRE is built using the default setting of **pre**, which was shown in (Fokkema, 2020) to provide a good trade-off between accuracy and interpretability.

- RandomForest (RF) are built using the R-package **randomForest** (Breiman, 2001). The number of features sampled at each split is left to default ($\lfloor\sqrt{p}\rfloor$) and normal bootstrapping used for resampling. RF is used as a out-of-the-box baseline.

- XGBoost (XGB) is tuned via Bayesian Optimization, as it relies much more on suitable parameters, which is done with the R-package **mlrMBO** (Bischl et al., 2018). The learning rate is considered between $[0.005, 0.1]$, covariates per tree between $[0.7, 1]$, subsample per tree between $[0.2, 1]$ and the *maxdepth* of trees as $\{1, 2, 3, 4, 5, 6\}$. The budget is set to 20 and the remaining values to default.

- The Generalized Rule Model (GLR) was built using the **aix360** python library. For the tunable parameter values $\lambda_0 \in \{0.001, 0.005, 0.01, 0.03, 0.05, 0.07, 0.09\}$ were considered and the λ_0 chosen that maximises the brier score in a 5-fold cross-validation.

- SIRUS was build using the **sirus** R-package and using the **sirus.cv** routine to determine the optimal number of rules for the final ensemble (maximizing accuracy).

- We also tested another XGBoost based RuleFit version, implemented in the R-package **xrf**, which was omitted in the main manuscript for space limitations, but is shown in the following. The XRF model is run with default settings and might lead to better results if tuned.

C Additional Results

C.1 Graphical Representation of the Deviation in AUC.

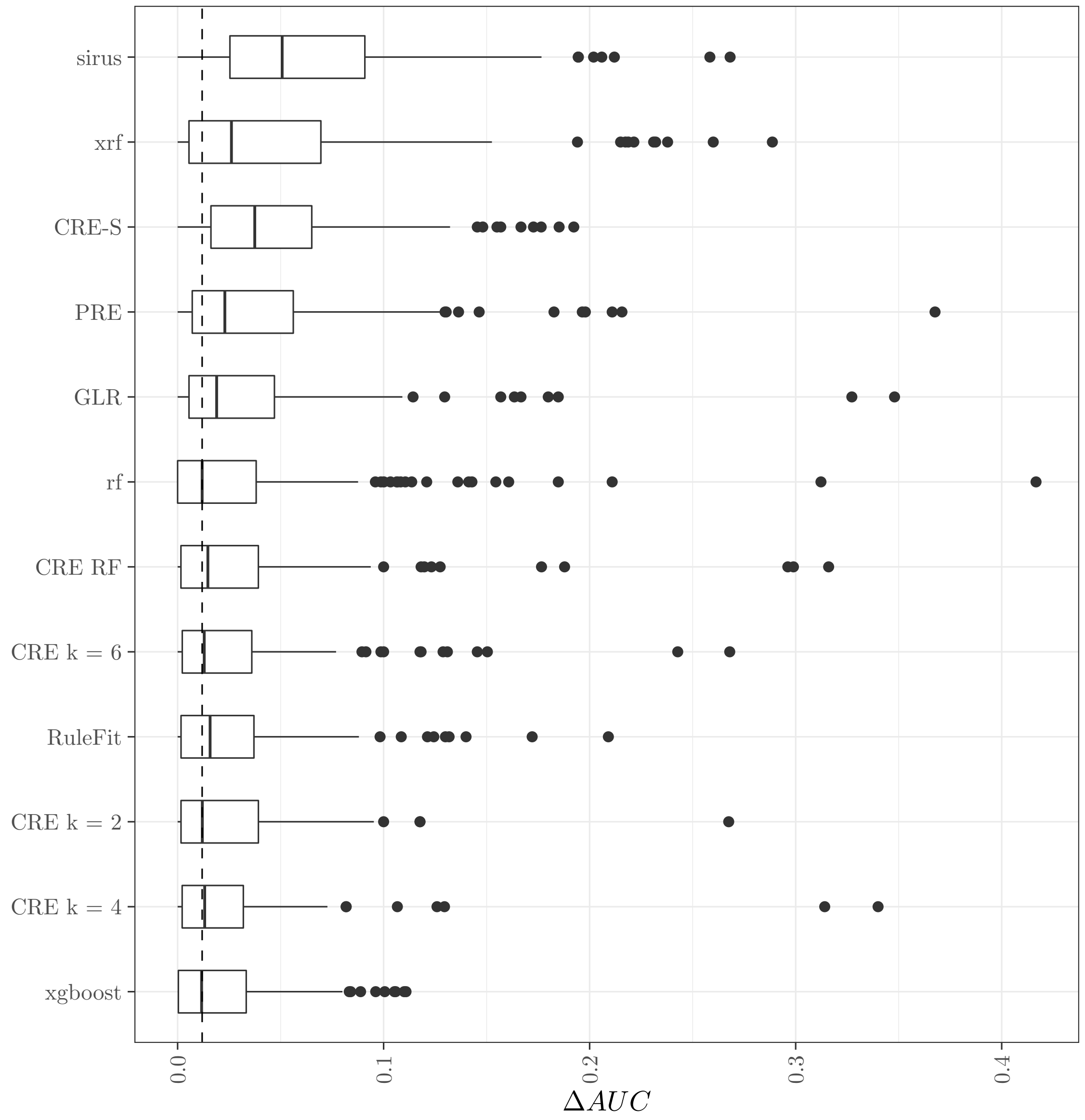


Figure 1: Graphical Representation of the results presented in Section 5. Shown are the boxplots of ΔAUC (cf. Section 5) over the different folds and datasets, ordered by their median ΔAUC.

Figure 1 implies an overall stable and good performance of the CRE based model.

C.2 Graphical Representation of the number of coefficients

Figure 2 shows the distribution of the number of coefficients. CRE clearly outperforms RuleFit and some versions of CRE are very competitive to SIRUS and GLR (while preserving a better AUC).

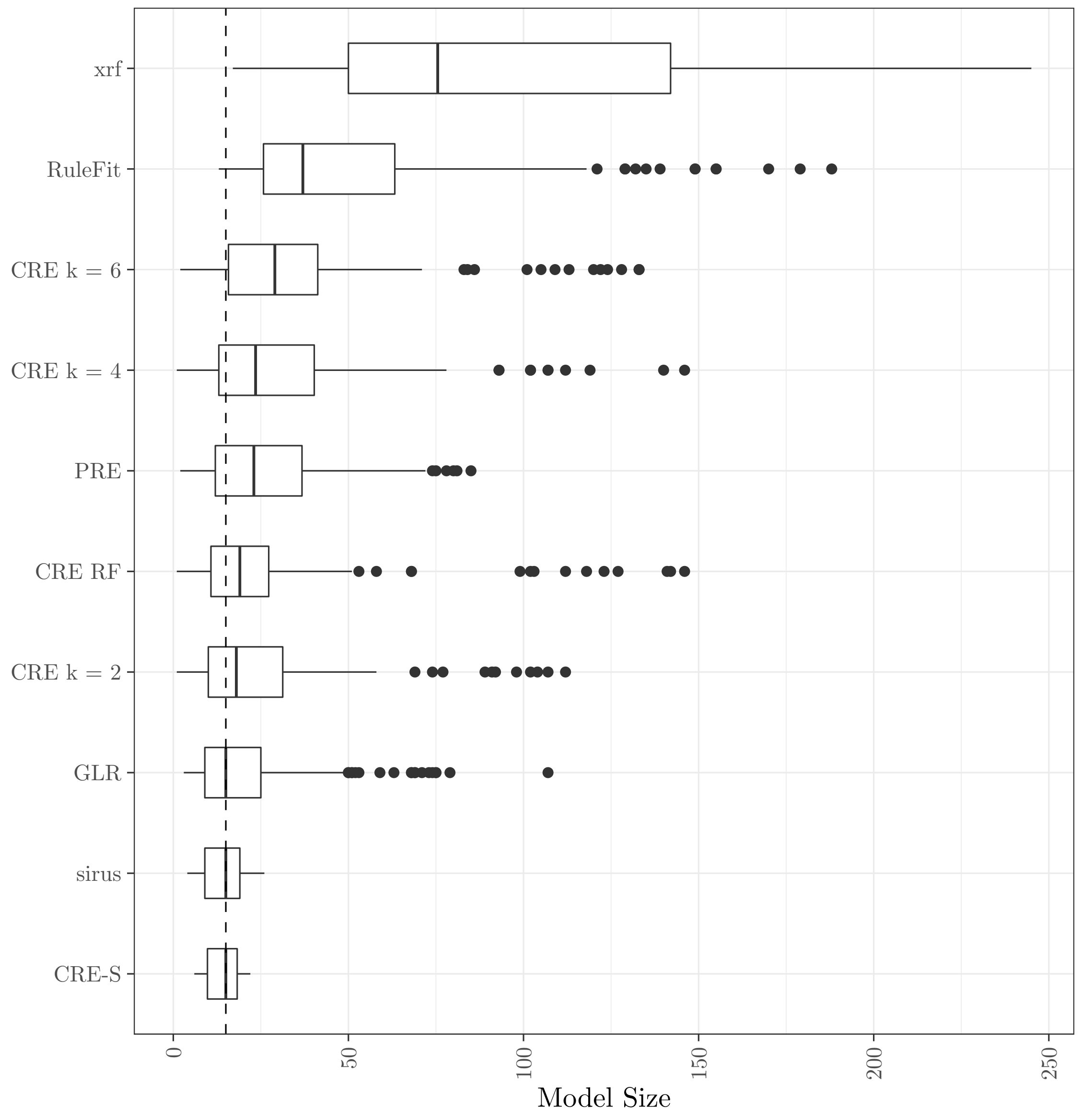


Figure 2: Graphical Representation of the results presented in Section 5. Shown are the boxplots of ΔAUC (cf. Section 5) over the different folds and datasets, ordered by their median ΔAUC.

C.3 Accuracy

Table 1: Results in Terms of accuracy.

dataset	CRE_S	$CRE_{k:2}$	$CRE_{k:4}$	$CRE_{k:6}$	CRE_{RF}	PRE	RF	RuleFit	SIRUS	XGB	XRF
Australian	0.841	0.864	0.862	0.868	0.864	0.861	0.865	0.870	0.829	0.868	0.854
Banknote	0.905	0.999	0.999	0.999	0.999	0.989	0.993	0.996	0.899	0.998	0.996
Biodeg	0.663	0.855	0.860	0.866	0.862	0.855	0.868	0.873	0.767	0.863	0.855
Blood Transf	0.762	0.765	0.761	0.762	0.767	0.762	0.751	0.781	0.762	0.789	0.773
Diabetes	0.651	0.776	0.758	0.752	0.762	0.752	0.768	0.769	0.698	0.758	0.734
Haberman	0.735	0.735	0.732	0.735	0.735	0.735	0.725	0.712	0.735	0.732	0.722
Heart	0.786	0.822	0.802	0.815	0.819	0.785	0.809	0.829	0.822	0.838	0.759
ILPD	0.714	0.715	0.714	0.705	0.722	0.714	0.705	0.696	0.714	0.700	0.703
Ionosphere	0.840	0.920	0.932	0.935	0.937	0.937	0.934	0.920	0.883	0.926	0.940
Liver	0.569	0.609	0.615	0.612	0.621	0.583	0.539	0.580	0.565	0.600	0.545
Parkinsons	0.809	0.897	0.912	0.907	0.907	0.856	0.902	0.902	0.866	0.922	0.913
Pop Failure	0.915	0.952	0.948	0.952	0.944	0.944	0.922	0.948	0.915	0.946	0.956
Sonar	0.732	0.857	0.842	0.838	0.856	0.785	0.842	0.847	0.756	0.842	0.837
Spambase	0.860	0.946	0.952	0.952	0.947	0.942	0.953	0.946	0.857	0.957	0.946
WBCD	0.939	0.967	0.967	0.961	0.967	0.963	0.960	0.965	0.942	0.970	0.959
Wilt	0.946	0.983	0.982	0.986	0.984	0.984	0.982	0.986	0.946	0.986	0.982
$\bar{r}$	9.312	4.344	5.594	4.500	3.812	6.906	6.750	5.031	8.812	4.031	6.906
ΔAUC	0.072	0.009	0.011	0.010	0.007	0.023	0.018	0.012	0.053	0.007	0.021

Although we believe accuracy to be less informative compared to AUC, we also provide tabular results of the accuracy (fraction of correctly classified samples, using $P(Y = 1) > 0.5$ as decision rule). The results are quite similar to the AUC results. Noteworthy difference is CRE_S which performs worse in terms of accuracy, implying that the prediction outputs are not well calibrated. Another noteworthy difference is, that using the accuracy as meassure, CRE_{RF} shows the overall strongest performance.

D Dataset Description of the Diabetes Data

To show the easy interpretability of CRE, we use in Section 4 the freely available Pima Diabetes data set. For a more detailed description, see (Smith et al., 1988). The full names of covariates are:

- `preg`: Number of times pregnant
- `plas`: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- `pres`: Diastolic blood pressure (mm Hg)
- `skin`: Triceps skin fold thickness (mm)
- `insu`: 2-Hour serum insulin (mu U/ml)
- `mass`: Body mass index (weight in kg/(height in m)^2) [2]
- `pedi`: Diabetes pedigree function
- `age`: Age (years)
- `y`: Class variable (0 or 1) (1 = Diabetes)

[2] Also referred to as BMI in the main paper, due to better understandability.