Compressed Rule Ensemble Learning

Preprint – Under Review

Malte Nalenz
Department of Statistics
Ludwig-Maximilians-University
Munich

Thomas Augustin
Department of Statistics
Ludwig-Maximilians-University
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, which reflects the marginal smoothing behaviour of the original ensemble. The final model, that we call *Compressed Rule Ensemble* (CRE), fits a linear combination of ensemble rules. On a variety of datasets we show empirically that CRE is both sparse and accurate, carrying over the ensemble behaviour, while remaining interpretable. CRE delivers predictive performance on par with state-of-the-art tree ensemble methods but with a model size that is substantially smaller compared to previous rule ensemble approaches. Predictions can be explained by looking at the active ensemble rules, which allows 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 analysing 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.

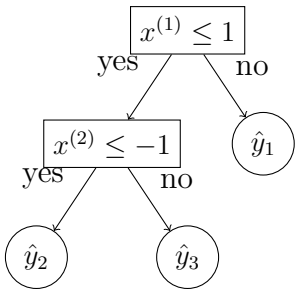
One approach towards interpretable machine learning models is to learn rule ensembles. As decision rules are composed of simple if-else statements, they are easier to interpret for humans, compared with deep decision trees. One such approach is RuleFit introduced by (Friedman and Popescu, 2008). Instead 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 promises 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, 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, that we call *ensemble conditions*. By averaging the discrete outputs of the individual conditions, ensemble conditions produce a smooth output, that 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 reflecting the uncertainty about the 'true' splitpoint. We argue that often already a few compressed rules allow to capture and interpret the central behaviour of the forest, allowing 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 different ways have been proposed to (greedily) induce decision rule ensembles. Classical approaches include the divide and conquer algorithms, that 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



$\phi(c)$	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.

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 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: the data is quantile transformed beforehand, to limit the possible splitpoints in trees, allowing to identify frequent pattern across trees. The most common decision rules are simply averaged to produce a prediction, without the need of a linear combination, which improves the model stability significantly.

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 (greedy) 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 its 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,

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

As the original forest contains a large number of rules, L1-penalization (Tibshirani, 1996) is used to shrink the large set of candidate rules down to the truly predictive ones.

4 Compressed rule ensembles (CRE)

As in previous versions of RuleFit, as 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 as is commonly featured in both random forests and (stochastic) gradient boosting, the split points 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 (greedy) tree induction algorithm will often chose 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. 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. The goal is therefore to preserve smoothness, but reshape it in a form that is accessible for human interpretation.

4.2 Clustering of similar conditions

To preserve the forests 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 cluster. The k centers in the k-means algorithm are chosen to minimize the intra-cluster variation,

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

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 clusterlabels for the splitpoints and $\mu = (\mu_1, ..., \mu_k)$ 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, that 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, as it results in an implicit weighting of regions found important by the forest method. As the appropriate number of clusters for each covariate is unknown a-priori, 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

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

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 UCI repository (Dua and Graff,

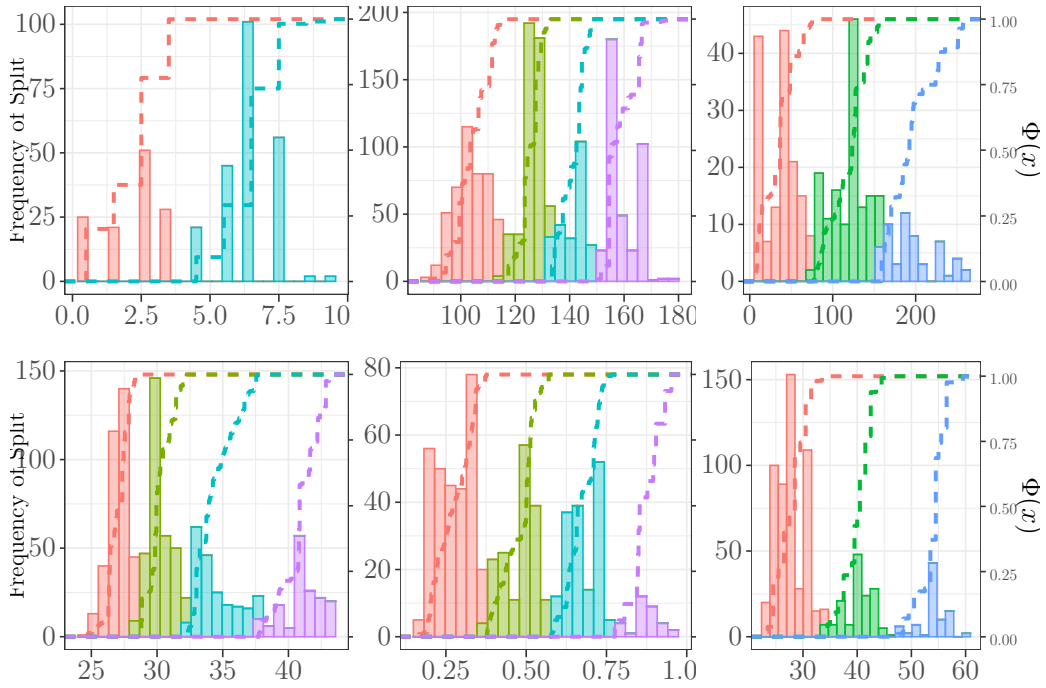


Figure 2: Distribution of splitpoints for the 6 most used covariates in the diabetes data set, using $k_{max} = 4$. Colours indicate the cluster solution from the *Ckmeans.1d.dp* algorithm. The dashed line shows the soft output $\Phi(x)$ for each cluster (compressed condition).

2017) and the clustering result using $k_{max} = 4$. We can see that the $\Phi(x)$ can follow arbitrary distributions. Also note the dense regions, forming clusters of splitpoints that are interesting for predictions. Lastly, if the underlying relationship is in fact a stepfunction, we expect the forest method to also be able to capture it and in return, the intervals of the clusters will 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

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

As all conditions in each cluster have the same output for any given x_i, this allows to 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.

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

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

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

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

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

$$\{\alpha^*, \beta^*, \beta_0^*\} = \arg\min_{\beta_0,\beta,\alpha} \left[L(y, F(x)) + \lambda \left(\sum_{j=1}^{p} |\beta_j| + \sum_{h=1}^{H} |\alpha_h| \right) \right], \tag{10}$$

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 for CRE.

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

$$\mu_{+,h} = \frac{1}{\sum_{i=1}^{N} \mathcal{R}_h(x_i)} \sum_{i=1}^{N} y_i \mathcal{R}_h(x_i), \tag{11}$$

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

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

$$\hat{y}_{i,h} = \mathcal{R}_h(x_i, v, g) \cdot \mu_{+,h} + (1 - \mathcal{R}_h(x_i, v, g)) \cdot \mu_{-,h}. \tag{13}$$

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 normal 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 whole tree ensemble can be summarised by a small number of ensemble rules.

4.5 The effect of of ensemble compression

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. We found a relatively small value of k_{max} (e.g. $k_{max} = 4$) is usually a good choice as discussed below.

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 (10) 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. On datasets where the number of predictive covariates is relatively small, as is in the Diabetes data, around 60 % of $\mathcal{R}_h$ can be removed from the initial set (with $k_{max} = 4$). If the splits distribute more evenly over a large number of covariates, the reduction is still notable but less pronounced.

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. R-code to reproduce all results will be made available upon publication.

5.1 Experimental setup

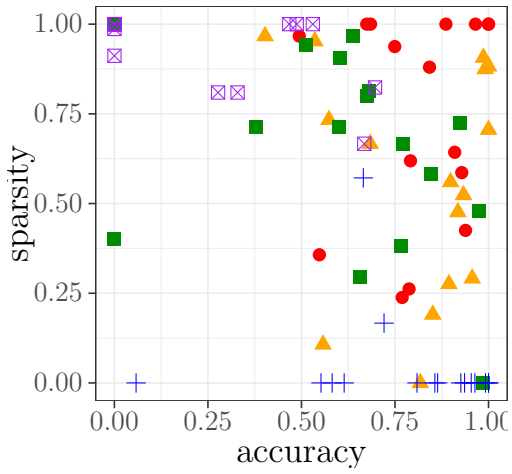
For comparison we use 16 binary classification datasets from the UCI repository (Dua and Graff, 2017). We chose datasets that consist of mostly numerical covariates and require minimal preprocessing. A detailed description of selection criteria and preprocessing, algorithm settings and additional results 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).

Table 1: Accuracy measured in AUC for the competing methods on the 16 benchmark datasets.

dataset	CRE_S	$CRE_{k:2}$	$CRE_{k:4}$	$CRE_{k:6}$	CRE_{RF}	PRE	RF	RuleFit	SIRUS	XGB
Australian	0.901	0.937	0.939	0.944	0.935	0.930	0.936	0.938	0.921	0.939
Banknote	0.986	1	1	1	1	1.000	1.000	1.000	0.972	1
Biodeg	0.871	0.929	0.931	0.930	0.919	0.915	0.938	0.924	0.840	0.932
Blood Transf	0.732	0.730	0.750	0.748	0.741	0.724	0.668	0.751	0.708	0.746
Diabetes	0.823	0.830	0.830	0.831	0.832	0.829	0.825	0.840	0.807	0.841
Haberman	0.712	0.683	0.670	0.680	0.621	0.676	0.685	0.708	0.651	0.688
Heart	0.898	0.910	0.896	0.896	0.888	0.877	0.905	0.897	0.899	0.906
ILPD	0.709	0.728	0.723	0.718	0.704	0.735	0.752	0.706	0.729	0.724
Ionosphere	0.955	0.963	0.964	0.965	0.954	0.965	0.981	0.968	0.941	0.970
Liver	0.649	0.666	0.679	0.644	0.667	0.623	0.564	0.657	0.644	0.654
Parkinsons	0.906	0.945	0.959	0.950	0.968	0.857	0.953	0.960	0.888	0.962
Pop Failure	0.907	0.947	0.945	0.952	0.946	0.947	0.920	0.925	0.889	0.946
Sonar	0.863	0.923	0.927	0.910	0.925	0.875	0.949	0.915	0.829	0.940
Spambase	0.963	0.985	0.986	0.986	0.985	0.980	0.987	0.985	0.933	0.988
WBCD	0.991	0.992	0.993	0.991	0.993	0.992	0.992	0.989	0.981	0.995
Wilt	0.952	0.990	0.992	0.991	0.993	0.991	0.990	0.993	0.901	0.990
Mean Rank	7.688	4.875	4.125	4.750	5.188	7	4.875	4.812	8.750	2.938
Mean ΔAUC	0.033	0.012	0.011	0.014	0.018	0.027	0.019	0.012	0.051	0.008

Table 2: 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), PRE (green square), RuleFit (blue cross), SIRUS (purple squares).

dataset	$CRE_{k:2}$	$CRE_{k:4}$	$CRE_{k:6}$	CRE_{RF}	PRE	RuleFit	SIRUS
Australian	18	26	32	19	20	40	15
Banknote	11	21	29	18	45	45	14
Biodeg	52	62	69	51	50	106	22
Blood Transf	7	8	8	6	9	22	6
Diabetes	9	18	23	10	28	36	9
Haberman	2	3	3	2	4	23	6
Heart	13	17	18	18	22	28	18
ILPD	10	10	15	9	10	68	8
Ionosphere	33	40	43	22	23	27	15
Liver	7	9	10	8	8	24	10
Parkinsons	22	25	27	20	14	35	18
Pop Failure	29	38	40	27	25	46	17
Sonar	50	61	61	42	31	54	19
Spambase	95	112	119	121	75	149	22
WBCD	31	32	36	24	28	36	15
Wilt	16	23	29	23	55	91	17
Mean Rank	2.75	4.44	5.47	2.88	3.81	6.62	2.03



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., 2017) 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 treedepth 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 and implemented in the package.

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 or η, are tuned. Better predictive performance may be reached, but in this article we are interested in the 'out-of-the-box' performance. 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 on each dataset the average number of rules used by SIRUS, leading to the overall same model complexity as SIRUS. We expect the same number of ensemble rules to generalize better compared to normal rules.

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 and average deviation from the best AUC value over all datasets. 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. It is interesting to note that $CRE_{k:2}$ is still competitive in terms of accuracy to random forest and RuleFit and outperforming most of the other rule based competitors. Using a higher compression parameter $CRE_{k:6}$ is not beneficial in the analysed datasets. This can be contributed to the regularizing effect of ensemble compression, making $k_{max} = 4$ our recommended default choice for prediction. SIRUS, CRE_S and PRE are on average less accurate. As seen by the average deviation from the best method, CRE models and RuleFit are on average not much behind the best method, implying a stable performance. The same is not true for SIRUS, CRE_S and PRE, which sacrifice on average a notable amount of accuracy.

Model Complexity Table 2 shows the number of selected rules and linear terms. In terms of model complexity SIRUS, $CRE_{k:2}$, CRE_{RF} and PRE produce the smallest models, with SIRUS being the winner. However, as discussed above, SIRUS and PRE have to sacrifice an substantial amount of accuracy to achieve this goal, whereas $CRE_{k:2}$ on most datasets produces similarly sparse models, while remaining competitive in predictive performance. $CRE_{k:4}$ takes in slightly more rules, but also produces reasonably small models on most datasets, whereas also showing strong accuracy. RuleFit produces overall the largest models.

Accuracy vs. Sparsity We conclude that CRE produces models that are both accurate and sparse, whereas all of the competing methods have to compromise either

Table 3: Model output for the Diabetes data.

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

aspect. This trade-off can be seen in Figure 3, where only $CRE_{k:2}$ and $CRE_{k:4}$ are able to consistently achieve good accuracy and sparsity (top right quadrant). This is enabled through the usage of ensemble rules, that allow smooth decision boundaries even for extremely 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, when combined with gradient boosting, while producing more sparse solution, when using random forest to generate the rules. While in this study a well tuned XGBoost model is the most accurate, CRE is on average not much worse while producing very well interpretable models.

5.3 Interpretation

Finally, we showcase how CRE can be used for vivid interpretation. Here we focus on the literal interpretation of the rules, as they are the main advantage of rule ensembles. The following table shows an output of $CRE_{k:4}$ for the Diabetes dataset [1]:

It becomes immediatly obvious that diabetes is strongly connected to age and BMI and its interaction. BMI appears to be the most important covariate, appearing in almost all of the ensemble rules, often in combination 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, but no single split value would describe the relationship well and would be rather arbitrary. The distribution of split points and $\Phi(x)$ can be visualised, as shown in Figure 2.

CRE can also give an explanation of how a prediction is produced. Table 4 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

[1] A description of the covariates can be found in the supplementary material. Y = 1: diabetes positive.

Table 4: CRE prediction explanation.

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

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 split points in the ensemble rule fire, leading to an increase in risk of $\exp(0.21) = 1.24$. If instead hard rules were used, the rule could only give the full risk increase or none at all. 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 are interesting for interventions.

6 Conclusion and Future 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. We argue, that the increase in complexity, due to smooth decision rules, does not harm interpretability. On the contrary, it resembles human intuition, so that the interpretation reflects the models uncertainty better.

We expect CRE to be more stable than RuleFit, as the ensemble rules are less dependent 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. The approach in (Bénard et al., 2021) requires discretizing the data, which does not make sense for CRE. Suitable stability measures would be highly desirable, for future work.

Compressed rules may also be interesting to approximate a forest by means of a simpler model. To this end, rule compression can be used 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.

Supplement 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 (**?**).

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

Supplement 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., 2017). 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.

Supplement C – Additional Results

The following graphs shows a graphical visualisation of the results presented in section 4 of the paper:

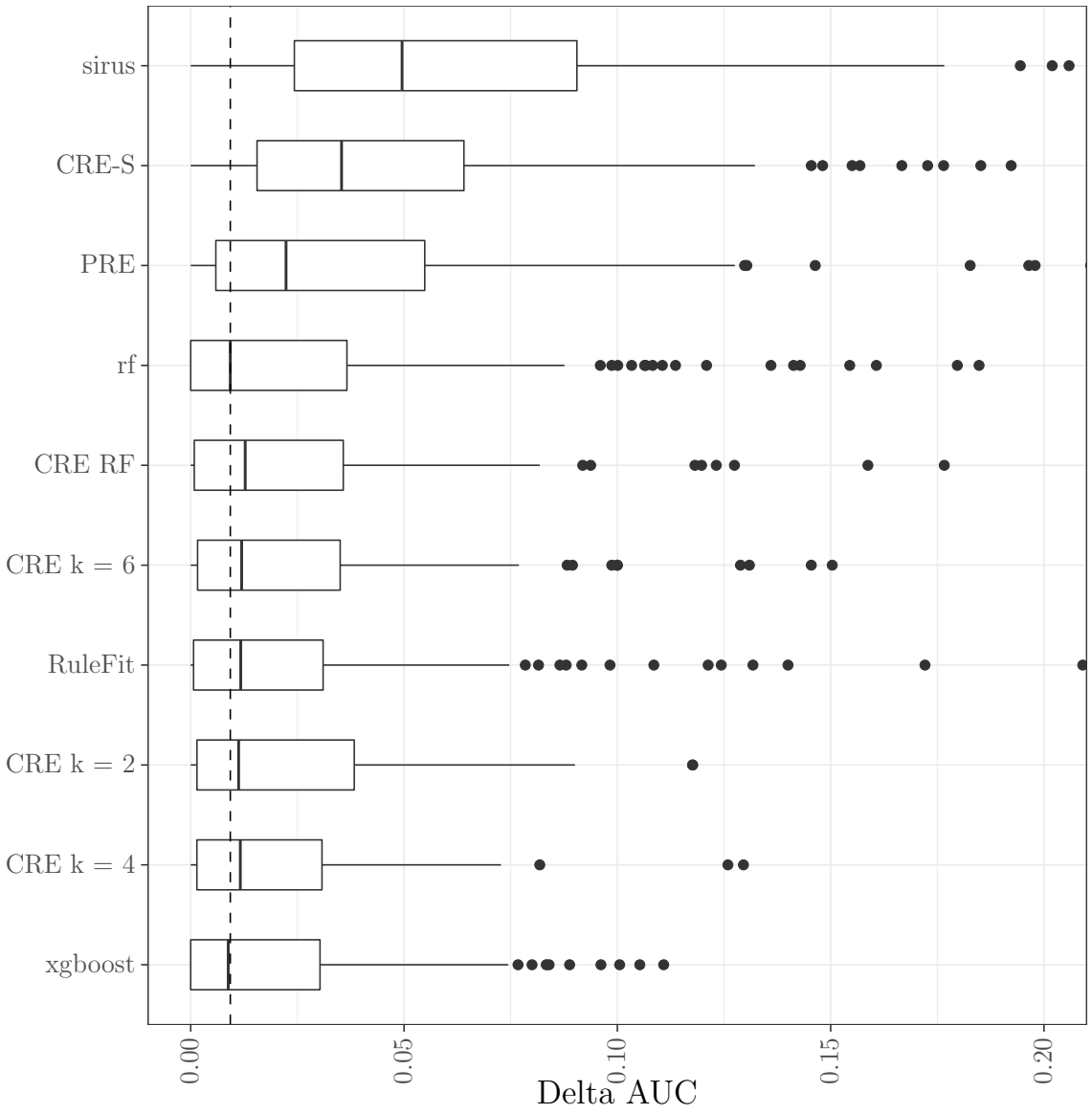


Figure 3: ΔAUC for the competing methods. The best performing method will have $\Delta AUC = 0$ in the given fold. The methods are ordered by the mean ΔAUC.

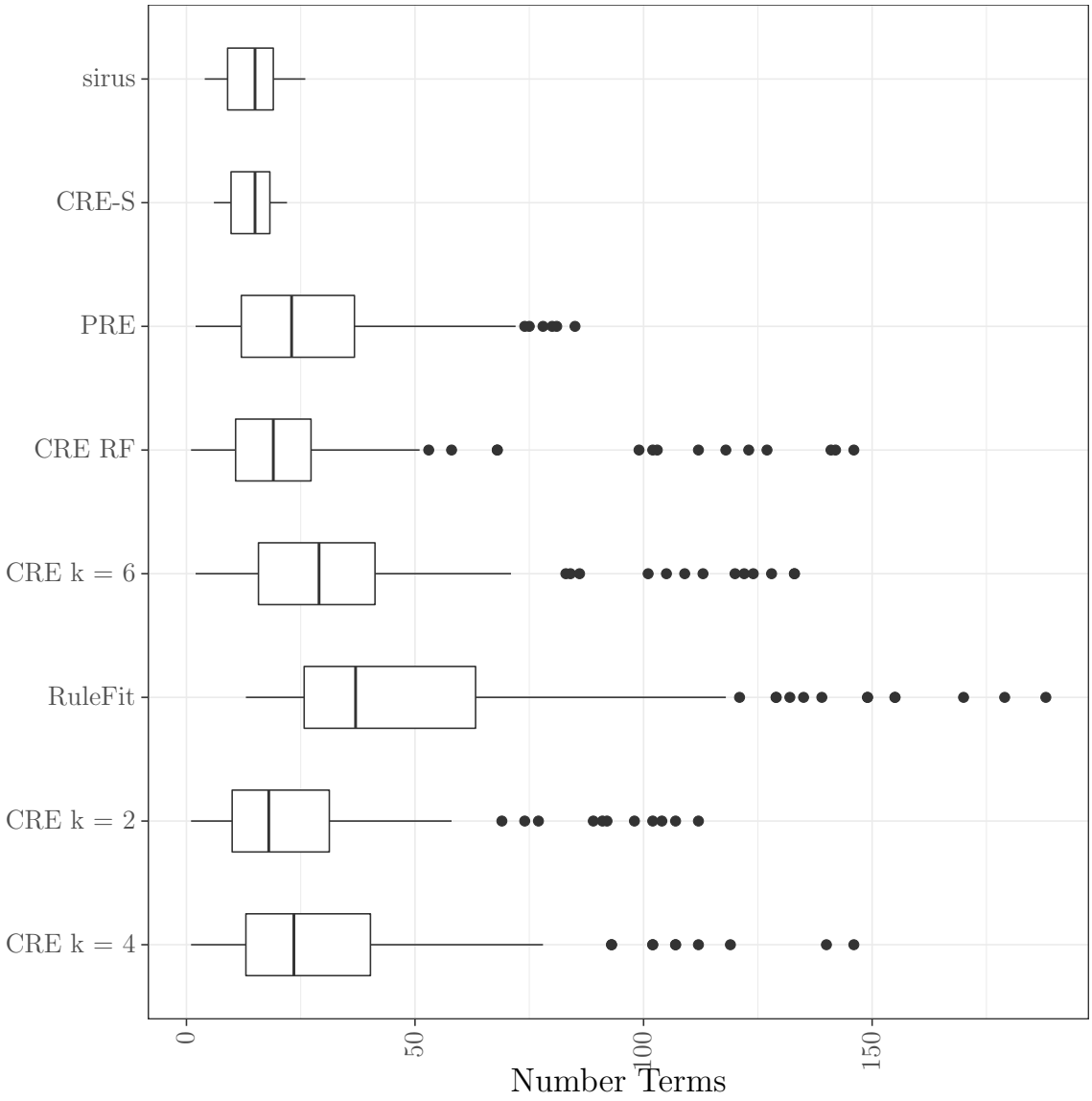


Figure 4: Number of non-zero coefficients (rules or linear terms).

Table 5: Performance meassured by accuracy over the 16 datasets.

dataset	CRE-S	$CRE_{k:2}$	$CRE_{k:4}$	$CRE_{k:6}$	CRE_{RF}	PRE	RF	RuleFit	SIRUS	XGB
Australian	0.841	0.864	0.862	0.868	0.864	0.861	0.865	0.870	0.829	0.868
Banknote	0.905	0.999	0.999	0.999	0.999	0.989	0.993	0.996	0.899	0.998
Biodeg	0.663	0.855	0.860	0.866	0.862	0.855	0.868	0.873	0.767	0.863
Blood Transf	0.762	0.765	0.761	0.762	0.767	0.762	0.751	0.781	0.762	0.789
Diabetes	0.651	0.776	0.758	0.752	0.762	0.752	0.768	0.769	0.698	0.758
Haberman	0.735	0.735	0.732	0.735	0.735	0.735	0.725	0.712	0.735	0.732
Heart	0.786	0.822	0.802	0.815	0.819	0.785	0.809	0.829	0.822	0.838
ILPD	0.714	0.715	0.714	0.705	0.722	0.714	0.705	0.696	0.714	0.700
Ionosphere	0.840	0.920	0.932	0.935	0.937	0.937	0.934	0.920	0.883	0.926
Liver	0.569	0.609	0.615	0.612	0.621	0.583	0.539	0.580	0.565	0.600
Parkinsons	0.809	0.897	0.912	0.907	0.907	0.856	0.902	0.902	0.866	0.922
Pop Failure	0.915	0.952	0.948	0.952	0.944	0.944	0.922	0.948	0.915	0.946
Sonar	0.732	0.857	0.842	0.838	0.856	0.785	0.842	0.847	0.756	0.842
Spambase	0.860	0.946	0.952	0.952	0.947	0.942	0.953	0.946	0.857	0.957
WBCD	0.939	0.967	0.967	0.961	0.967	0.963	0.960	0.965	0.942	0.970
Wilt	0.946	0.983	0.982	0.986	0.984	0.984	0.982	0.986	0.946	0.986
Mean Rank	8.562	4.031	5.281	4.250	3.562	6.469	6.312	4.625	8.062	3.844
Delta Best	0.071	0.009	0.011	0.010	0.007	0.022	0.018	0.011	0.053	0.007

Although we believe accuracy to be less informative compared to AUC, we also provide tabular results of the accuracy. The results are quite similar to the AUC results. Noteworthy difference is $CRE-S$ which performs worse when meassured in 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.

Supplement 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 (**?**). 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.

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.

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

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. (2017). *mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions*.

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.

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 *Proceedings of the 25th International Conference on Machine Learning*, pages 224–231.

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*, volume 96, 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 *Proceedings of the 28th 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.

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.

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

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 *Proceedings of the 17th International Conference on Machine Learning*, pages 1135–1142.