Master Thesis in Statistics and Data Mining

Horseshoe RuleFit – Learning Rule Ensembles via Bayesian Regularization

Malte Nalenz

Division of Statistics and Machine Learning
Department of Computer and Information Science
Linköping University

June 2016

Supervisor:
Mattias Villani
Division of Statistics and Machine Learning
Linköping University

Examiner:
Oleg Sysoev
Division of Statistics and Machine Learning
Linköping University

Abstract

I propose Hs-RuleFit, a learning method for regression and classification, which combines rule ensemble learning based on the RuleFit algorithm with Bayesian regularization through the horseshoe prior. To this end theoretical properties and potential problems of this combination are studied. A second step is the implementation, which utilizes recent sampling schemes to make the Hs-RuleFit computationally feasible. Additionally, changes to the RuleFit algorithm are proposed such as Decision Rule post-processing and the usage of Decision rules generated via Random Forest.
Hs-RuleFit addresses the problem of finding highly accurate and yet interpretable models. The method shows to be capable of finding compact sets of informative decision rules that give a good insight in the data. Through the careful choice of prior distributions the horseshoe prior shows to be superior to the Lasso in this context. In an empirical evaluation on 16 real data sets Hs-RuleFit shows excellent performance in regression and outperforms the popular methods Random Forest, BART and RuleFit in terms of prediction error. The interpretability is demonstrated on selected data sets. This makes the Hs-RuleFit a good choice for science domains in which interpretability is desired.
Problems are found in classification, regarding the usage of the horseshoe prior and rule ensemble learning in general. A simulation study is performed to isolate the problems and potential solutions are discussed.
Arguments are presented, that the horseshoe prior could be a good choice in other machine learning areas, such as artificial neural networks and support vector machines.

Acknowledgments

I like to thank Mattias Villani for the great supervision. You gave me a lot of freedom to try out ideas and yet took your time to discuss them with me. The discussions were always helpful and inspiring. I like to thank my companions Han, Miriam, Hector, Kostas and Maria. Our coffee breaks helped me to keep my sanity in this not always easy times. I finally like to thank Olivia for always believing in me. You always make me smile after the long days.

Contents

Abstract **iii**

Acknowledgments **iv**

Contents **v**

List of Figures **vi**

List of Tables **vii**

1 Introduction **1**

2 Theory **4**
2.1 Supervised Learning 4
2.2 Ensemble Learning 5
2.3 Rule Ensembles 7
2.4 RuleFit 7
2.5 Bayesian Regularization with the Horseshoe prior 10

3 Method **16**
3.1 Implementation of the Rule Generating Process 16
3.2 Implementation of the Horseshoe Regression 19
3.3 Posterior Summary 27

4 Results **29**
4.1 Regression 29
4.2 Classification 33
4.3 Applications 35
4.4 Simulation Study -Rule Ensembles in Classification 40

5 Discussion **43**
5.1 Results and Implications 43
5.2 Method 44
5.3 Implications for other areas 47

6 Conclusion **48**

Bibliography **51**

List of Figures

2.1 Decision Tree for the Boston Housing data. 8
2.2 The prior distribution of the shrinkage coefficient κ under the Horseshoe prior and the Lasso (Double-Exponential). $\kappa = 0$ implies no shrinkage, $\kappa = 1$ total shrinkage. 11
2.3 Difference between Horseshoe prior and Lasso in shrinking uninformative covariates. 13
2.4 Posterior densities. The Horseshoe posterior leads to bimodal distributions, with either x1 or x3 equal to zero. The Lasso more unimodal with mode in between the horseshoe mode. Under Horseshoe uncorrelated predictor x2 not effected by multicollinearity. 14

3.1 Monte Carlo simulated prior density $p(\beta)$ for $\tau \sim C^+(0,1)$(standard) and $\tau \sim C^+(0,0.01)$(robust). The robust prior puts less mass on unreasonable values. 24
3.2 The hyperparameter A_l depending on the support of a rule and its length. The function is slowly decaying to give complicated and specific rules still a non-zero prior probability of being used. 26

4.1 Boxplots showing the RRMSE across the 160 train/test partitionings over the 16 data sets. There is still a considerably amount of values higher than 3 for Random Forest (12.5%) and BART (6.1%) 31
4.2 The Traceplot for the 9 largest coefficients of Hs-Rulefit on the Boston Housing data. 35
4.3 RuleHeat for the Boston Housing Data. 36
4.6 RuleHeat for the Prostate Cancer Data set. 38
4.9 The generated data. It is not separated, but complete separated subspaces can be found by Decision Trees. 41
4.10 Comparison of the predicted probabilities of the different methods, darker blue represents probabilities close to zero, while lighter blue indicates higher probabilities. Red points are $y = 0$ yellow are $y = 1$. 42

List of Tables

2.1 Corresponding rules, defining the Decision Tree. 8

4.1 The Settings for the used methods. 30
4.2 The 16 Regression Data sets. 30
4.3 10-fold Cross validation results over the 16 regression data sets. Each entry shows the RMSE and in parentheses the rank on this data set. The best result is marked in bold. 32
4.4 The eight classification data sets used for evaluation. 33
4.5 10-fold Cross validation results over the 8 classification data sets. Each entry shows the average misclassificationrate and in parentheses the rank on this dataset. The best result is marked in bold. 33
4.6 10-fold Cross validation results on the Prostate Cancer data set. 38

1 Introduction

> Everything should be as simple as possible, but not simpler.
>
> Albert Einstein

In the last decade, statistical learning became increasingly important in solving real world problems. Enabled through computational power it found novel applications in a variety of areas, which were too difficult to handle in the past. Social media analysis, image recognition, medical diagnostics, prediction of costumer behaviour or political election outcomes are only some examples where statistical methods and machine learning have been applied succesfully. In fact, it is hard to think of a domain in which statistical learning is not applied today. As diverse as the topics are the goals of the applications. In image recognition pure predictive accuracy is the main focus, while in medical diagnostics the question is governed by finding the underlying mechanisms of a disease. This information can be used in further research, such as the design of new drugs.

Machine learning research has traditionally focused mainly on the predictive performance of algorithms. The advent of Random Forest and Boosting and the reinterest in Support Vector Machines and Neural Networks are only some of the most influenctual developments. These algorithms achieve state of the art predictive performance in most learning problems, however the gain in accuracy comes at the cost of an increased model complexity. All four algorithms share the characterstic that they produce to a certain degree black box models. It is not clear what a prediction is based on neither is the model able to describe the underlying relationship. An often stated result is, that the most powerful models are useually the least interpretable ones [39].

Kuhn & Johnson[39] suggest an inverse relationship between model accuracy and interpretability. They conclude that predictive performance should be the main objective. To a certain degree this makes sense. One should be cautious about a model that is not able to make accurate predictions. This view can be traced back to the philosophical argument that the goodness of a scientific theory is determined by its ability to make risky predictions [40]. It is always easy to find ad-hoc explanations that describe perfectly the seen data. On the other hand, if a model is able to constantly give correct predictions it is likely that it uses the correct underlying relationship or at least approximates it. By that the model generalizes

well. The questions remain to what situations or data the model generalizes. This problem is refered to as *concept drift*. The relationship might be different in another population or change over time. Therefore a requirement in most natural sciences for a model is, that it is able to give the mechanism which lead from a set of conditions to an output [60]. This allows judgement about the validity and consisticency of the found model. In fact for most science domains the main focus lays on finding the underlying mechanisms. Examples are the social sciences, economy or physics. Developing algorithms which achieve predictive accuracy on the cost of interpretability means excluding those fields from using it.
Another field where interpretability is a major concern is medical data. High accuracy in predicting a disease from genetic expressions is important to give confidence in a model, but is typically not a mean in itself. Most important is the finding of meaningful patterns, for example of genes-disease associations. The identification of the underlying mechanism enables the design of medical tests, novel drugs [47] and to generate interesting research questions for experts in the field [62]. The same goes for the field of personalized medicine: not many patients and doctors will trust a diagnosis without an explanation what the prediction is based on [41].
Recent developments in machine learning suggest that there is a "sweet spot" between accuracy and interpretability. Letham et al. [41] introduced a decision list, based on a small set of simple if-else rules that achieves reasonable accuracies. Their stroke prediction decision list heavily outperforms the theory driven previous diagnostic tool, while being equally easy to understand. A similar appraoch for gene expression data showed promising results in predicting cancer based on a small number of decision rules [33].
In current machine learning research there has been a considerably effort in the simplification of tree enesembles. Reguralized Greedy Forest [37], Nodeharvest [45] and ISLE [26] use regularization to limit the complexity of Ensemble of Tree models. In their results regularization *increases* predictive performance while limiting the model complexity. This questions the stated inverse relationship between accuracy and model complexity.
In between those two approaches is RuleFit [27] which simplifies an Ensemble of Trees to a set of simple if-else rules. While showing promising results follow-up studies report instability [46]. Also the set of rules is often still too large for easy interpretation [41]. However the idea is tempting and strikingly simple. Given an Ensemble of Tree model decompose it into its defining decision rules and use regularization to remove unnecesary rules. Decision Rules are inherently easy to interpret, hence the resulting model promises to be highly accurate and easy to understand.
Regularization plays a key role in the above methods and an inappropriate choice might lead to suboptimal solutions. The above methods utilize the least absolute shrinkage and selection operator (Lasso) [63] for regularization, which is in machine learning literature the default choice for sparsification problems. In the last years strong competitors to the Lasso have been proposed within the Bayesian framework. Park & Casella [50] formalized the bayesian Lasso, which leads to similar solutions as the ordinary Lasso but allows estimation of model uncertainty and bayesian model averaging [54]. A different approach are global-local shrinkage priors, with the horseshoe prior [15] the double-pareto prior [5] and the normal-gamma prior [34] as most popular representatives. They unite in the careful choice of prior distributions to satisfy desirable theoretical properties, such as efficient handling of noise and signal. The global-local shrinkage priors have been shown to be superior to the Lasso in many regression settings [5][15].
This work argues, that the RuleFit can be improved by utilizing bayesian global-local shrinkage priors, namely the horseshoe prior as a good default choice [15], instead of the Lasso. The horsehoe prior offers theoretical properties which go perfectly together with the RuleFit idea. It removes uninformative predictors agressively, leading to more compact models. At the same time the horseshoe prior allows informative predictors to remain unshrunk. This is desirable in terms of interpretation and was also shown to reduce prediction error [15]. The expected result is that the horseshoe is able to find very compact rule sets only consisting of

the truly informative rules, which describe the most important mechanisms in a data set.
This work proposes the Horseshoe-RuleFit (Hs-RuleFit) for regression and classification that uses the horseshoe prior for regularization in conjunction with the rule ensemble idea of RuleFit. To this end first the theoretical properties of rule ensembles and the horseshoe prior are discussed in Chapter 2. Special focus lays on the specific requirements of rule ensemble learning and it is analyzed in a theoretical way if the horseshoe prior is capable to fulfill those requirements.
One major challenge in this work was the implementation, described in detail in Chapter 3. To handle the high dimensionality of rule ensemble learning an efficient Gibbs-Sampling scheme and recently proposed algorithms for speeding up the sampling procedure are used. The combination of rule ensemble learning and bayesian regularization poses new problems and possibilities. Bayesian learning allows to incorporate prior information in the model. This is in this work utilized to help the Hs-RuleFit overcome a main weaknesses of the original RuleFit, namely the overfitting on overly complicated rules, by incorporating the prior knowledge to prefer simple and general rules. The model is also extended to classification via different data augmentation schemes. Even though computationally straightforward, problems occured, which might be quite fundamental. In this context the problem of separation in conjunction with rule learning is discussed and changes to the horseshoe prior proposed in order to make the Hs-RuleFit applicable in classification.
The predictive performance is tested on a variety of real data sets for regression and classification in Chapter 4. Also evaluated is the interpretability of the produced models of Hs-RuleFit on a data set from the social science domain and a genetic-expression data set. For this the interpretational repertoire from the original RuleFit [26] is extended with novel visualization tools which exploit the decision rule structure. Also a simulation study is performed in order to understand the functioning of rule ensemble learning in classification better.
Limitations in this work are given by the fact, that both RuleFit and global-local shrinkage priors are farely new developments and not well understood yet. Both are designed in the regression context, while the extension to classification is not well studied. That means that in some parts this work is highly experimental, as no solid ground to built upon exists yet. The work is mainly delimited to the creation of a working baseline model, while the search for optimal choices has to be kept open for future work. Open problems, together with possible ways to adress these problems in future work are discussed in Chapter 5. This work also contributes to the question if the combination of machine learning and bayesian shrinkage prior methodology is fruitful. Regularization is widely used in machine learning and the results in this work therefore might also have implications for other areas of machine learning such as artificial neural networks and Support Vector Machines. It is argued in this work that the flexibility offered by the bayesian framework makes shrinkage priors a good choice, as they allow to tailor the regularization procedure to the machine learning problem at hand. However shrinkage priors also pose new problems of both theoretical and computational nature which need to be considered thoroughly.
Chapter 6 finishes with final remarks and implications of this work.

2 Theory

2.1 Supervised Learning

Supervised learning is the task of predicting future observations on the basis of past data. Usually only some aspect of the data is of interest. For example in genetic expression data the main interest lays in understanding the relationship between DNA-expressions and certain diseases.
Given a set of N past observations, for which we know the p-dimensional attribute vector $\boldsymbol{x}$ as well as the outcome y, $(x_i, y_i), i = 1, \ldots, N$, we seek to learn the underlying mechanism

$$y = F(x) + \epsilon, \tag{2.1}$$

where ϵ is an irreducible noise term which comes from random fluctuation as well as the effect of unobserved variables. This general framework is refered to as supervised learning, in contrast to unsupervised learning, in which the objective is typically to find groups of similar observations and the outcome is unknown.
The two major approaches to supervised learning are machine learning and statistical modeling. There is no clear demarcation line between those approaches, however they typically differ in the problem formulation. As this work borrows concepts from both areas it is worth to go into more detail.
In machine learning one useually starts with the definition of a Loss function $L(y, F(x))$ which meassures the lack of accuracy and seeks to minimize the predicion risk

$$R(F) = E_{x,y} L(y, F(x)). \tag{2.2}$$

The estimated approximation of the true underlying function $F(x)$ is then the solution to the minimization problem

$$F^*(x) = \arg\min_{F(x)} E_{x,y} L(y, F(x)). \tag{2.3}$$

As the distribution of y is unknown when doing prediction, equation (2.2) can only be estimated, useually done via cross validation. In this framework no assumptions about the distributions of x, y and $y|x$ are made. In fact many learners $F(x)$ are motivated from an optimizational point of view, rather than a probabilistic. Examples of widely used learners $F(x)$

are decision trees, neural networks, support vector machines and linear models. The latter is a good example to point out a difference between machine learning and statistics, as it can be motivated from both perspectives.
In the machine learning framework a linear model is based on the minimization of the mean squared error loss

$$L(y, F(x)) = (y - F(x))^2 \tag{2.4}$$

and

$$F(x) = x\beta. \tag{2.5}$$

Again no assumptions about the distributions are made. On the other hand the linear model can as well be cast from the statistical framework as

$$y|\boldsymbol{X}, \beta, \sigma^2 \sim \mathcal{N}(\boldsymbol{X}\beta, \sigma^2 \boldsymbol{I}_n). \tag{2.6}$$

If β and σ^2 are seen as fixed but unknown values, the objective is to maximize the likelihood of (2.6)

$$\hat{\beta}_{ML} = \arg\max_{\beta} f(y|\boldsymbol{X}, \beta, \sigma^2) \tag{2.7}$$

which is done by solving

$$\frac{\partial f(y|\boldsymbol{X}, \beta, \sigma^2)}{\partial \beta} = 0 \tag{2.8}$$

for β. In Bayesian statistics β and σ^2 are seen as unknown and therefore *random*. Instead of solving the maximization problem in equation (2.7) we are interested in the posterior distribution

$$f(\beta, \sigma^2|y, \boldsymbol{X}) \propto f(y|\boldsymbol{X}, \beta, \sigma^2) f(\beta|\sigma^2) f(\sigma^2). \tag{2.9}$$

F^* can then be defined as posterior mean, median or mode, or any other point estimate, depending on the goal. When the priors $f(\sigma^2)$ and $f(\beta|\sigma^2)$ are chosen uninformative the posterior mode is equivalent to the solution in (2.7). In this case the machine learning, the maximum likelihood and the bayesian apporach lead to the same solution, but this is by no means always the case. The solutions are based on very different theoretical foundations and there seems to be a certain aversion against combining them. For example Leo Breiman, the creator of Random Forest noted, that "when big, real, tough problems need to be solved, there are no Bayesians." Now that many computational restrictions are passed, Bayesian methodology was succesfully applied to many, large scale, problems. As in my opinion the bayesian approach is the most flexible as it allows to incorporate prior information, it is in this work used for statistical modeling. Also the necessary machine learning concepts in this work can be integrated in the (bayesian) statistical framework, while the opposite is not true.

2.2 Ensemble Learning

In supervised learing an often occuring problem is the overfitting of the training data. Overfitting is based on the Bias-Variance tradeoff. Reducing the Bias of a model by fitting every part of the variation in the training data is not always a good idea, as the model might capture random fluctuation rather than the true underlying function. This leads to poor generalization to new unseen data. When training a model, a natural question is how far we seek to reduce the Bias, hence fit to the training data.

There are many different approaches to find a good tradeoff for example Pruning in decision trees, early stopping in neural networks or regularization. One of the most powerful solutions is to combine a number of models f_l, $l = 1, \dots, m$ to an *ensemble*

$$F(x) = \alpha_0 + \sum_{l=1}^{m} \alpha_l f_l(x). \tag{2.10}$$

$f(x)$ is often refered to as weak learner. Equation (2.10) is only one way to combine models, an exhaustive review of ensemble methods is given in [55]. Note that (2.10) can also be interpreted as a generalized additive model (GAM), whith $f(x)$ as basis functions [24]. An often used analogy is to view $f(x)$ as experts, which come together as a commite. The influence of their opinion is given by the individual weight α_l, which is typically determined by some quality criteria of this experts opinion. Predictions of unknown outcomes are based on the weighted vote of the ensemble. Ensemble methods tend to work well, when the Bias of each individual base learner is low and the Variance between the base learners is high. That is, every base learner tries to explain a different aspect of the relationship between $\boldsymbol{x}$ and y. To stay in the experts analogy, the ensembles accuracy improves with the diversity of its opinions, which should build upon local knowledge. Also the more independent the opinions are from each other the better. While the individual weak learners might be only slightly better than random guessing, when blending the different opinions, the ensemble often generalizes very well to unseen data, as the prediction is not based on one single explanation, which might be only true for the training data.
Often utilized weak learners are decision trees. Decision trees are a highly adaptive method able to approximate any linear or non-linear relationship between x and y through recursive partitioning of the covariate space [61]. However the approximations of smooth functions, especially linear functions, can only be achieved through a huge number of recursive partitionings. One problem with the usage of decision trees is their tendency to overfit. The decision boundaries can change dramatically when only a single observation is added or removed from the training data. Interestingly in the framework of ensemble learning this is not a downside. The individual overfitting guarantees a diversity of the weak learners, if the local knowledge criteria is fulfilled. Local knowledge can be induced by sampling strategies, most prominent being aggregated bootstrapping (bagging) [10] and adaptive resampling (arcing) [11] [58]. Each tree is then grown on a different subset of the data, leading to a high diversity of trees.
Boosting and Random Forest can be both cast in this framework, however they differ in how the weak learners are created. Friedman & Popescu [26] proposed with ISLE a unified framework which covers the most popular tree based learning ensembles as special cases of an importance sampling procedure, see Algorithm 1.

Algorithm 1 ISLE Ensemble generation

1: $F_0(x) = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha)$
2: **for** $l = 1, \dots, k$ **do**
3: $\quad f_l = \arg\min_{f} \sum_{i \in S_m(\eta)} L(y_i, F_{l-1}(\boldsymbol{x}_i) + f(\boldsymbol{x}_i))$
4: $\quad F_l(\boldsymbol{x}) = F_{l-1}(\boldsymbol{x}) + \nu f_l(\boldsymbol{x})$
5: **end for**
6: ensemble = $\{f_l(\boldsymbol{x})\}_1^k$

$S_m(\eta)$ in line 3 denotes the usage of a sampling scheme with subsample size η and ν in line 4 is the memory, or shrinkage parameter, which determines at each step l how much the previous ensemble influences the next base learner $f_l(x)$. Different choices for base learners are possible, but in the following only Classification and Regression Trees (CART)[13] are considered for the Ensemble of Tree methods. Using non-parametric bootstrap with replacement

for $S_m(\eta)$, setting $\nu = 0$ and $L(y, f(x)) = (y - f(x))^2$ Algorithm 1 is equivalent to Random Forest [26] [12]. In the same manner the AdaBoost for classification [23] can be constructed via $L(y, f(x)) = exp(y \cdot f(x))$, $\nu = 1$ and $\eta = N$ and finally gradient boosting by modifying the Loss function appropriatly [26].

2.3 Rule Ensembles

A less prominent but steady tradition in machine learning research is to use decision rules as base learners. Decision rules can be seen as a special case of a decision tree, by setting the treedepth to one. Decision rules are also often referred to as decision (tree) stumps in this context. A property of decision rules is its easy human comprehensability. Each rule is defined by a simple if-else condition, which corresponds nicely with the human way of thinking. Decision trees on the other hand often consist of a very complex system of if-else conditions which to evaluate at the same time is difficult. Through their simple structure decision rules are also less prone to overfitting.
One approach to rule learning is called sequential-covering [28]. In each iteration a decision rule is induced, that tries to explain a set of observations, which are not covered any of the previous decision rules. After inducing a new rule all observations that are correctly classified are removed from the training set. Hence there is no overlaps, each observation is explained by exactly one decision rule. A classic sequential covering algorithm is RIPPER [18]. The disjunct set of rules allows very good interpretability. One problem is, that the underlying assumption that every observation is determined by exactly one rule is questionable. Imagine the situation where the goal is to predict a certain cancer given life-style predictors and genetic predictors. It is very likely that the probability of getting cancer is determined by different mechanisms, each potentially being a complicated combinations of lifestyle and genetic dispositions. Using only one decision rule per observation can not take this into account.
Another research area is to utilze decision rules in Boosting, most prominent in the early versions of AdaBoost [23] [57] and more recently ENDER [21] and SLIPPER [19]. Boosting allows to capture more complex relationships, as observations can be described by a number of rules in an additive fashion. In experiments this algorithms tend to outperform the sequential covering algorithms [21], however they lose in interpretability, as the ruleset often contain of thousands of rules, which also might be overlapping.
Rule Ensembles offer high human comprehensability, through their simple and intuitive structure. However the interpretability is highly dependent on the actual size of the ensemble. A set of thousands of rules, which have a similar strong influence on predictions becomes more or less a black box model. Therefore more compact rule sets are to prefer, if interpretability is the goal.

2.4 RuleFit

A new approach to rule learning was introduced by Friedman & Popescu [27] as RuleFit. It differs from traditional Rule Learning algorithms, in that it does not learn the rule ensemble directly in an iterative fashion. Instead it consists of a two step procedure. First candidate rules are generated with the tree ensemble methodology. Reason for this indirect approach is that tree ensembles have been shown to belong to the most accurate machine learning algorithms. This suggests that tree ensembles are capable of finding interesting subspaces, defined through decision rules. In rule learning the big searchspace of possible decision rules can be limiting. Exhaustive search for global optimal rules is not feasable for even medium scale problems. Using decision rules created by decision trees adresses this problem, as efficient algorithms are available here. It is possible that efficient direct rule induction algorithms will be found, but at this state the tree ensemble methods are generally superior in terms of

predictive performance [27]. The second step in RuleFit combines the found decision rules from the tree ensemble to an rule ensemble and performs a reweighting. As the tree ensemble methods use greedy search for constructing the tree, also the resulting rule set is not optimal. Instead it contains a lot of redundant and uninformative rules. Regularization is applied to melt down the set of rules to the truly informative ones. This is desirable both in terms of prediction and interpretability. The following two sections describe the two stages of RuleFit in detail.

2.4.1 From decision trees to decision rules

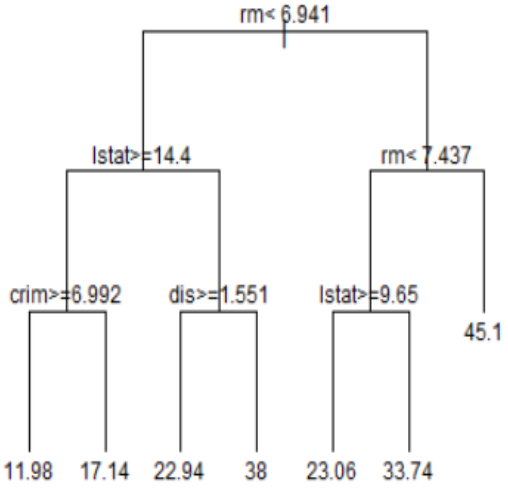


Figure 2.1: Decision Tree for the Boston Housing data.

Rules	Conditions
r_1	$rm \geqslant 6.94$
r_2	$rm < 6.94$
r_3	$rm < 6.94$ & $lstat < 14.4$
r_4	$rm < 6.94$ & $lstat \geqslant 14.4$
r_5	$rm < 6.94$ & $lstat < 14.4$ & $crim < 6.9$
r_6	$rm < 6.94$ & $lstat < 14.4$ & $crim \geqslant 6.9$
r_7	$rm \geqslant 6.94$ & $lstat < 14.4$ & $dis < 1.5$
r_8	$rm \geqslant 6.94$ & $lstat < 14.4$ & $dis \geqslant 1.5$
r_9	$rm \geqslant 6.94$ & $rm < 7.45$
r_{10}	$rm \geqslant 6.94$ & $rm \geqslant 7.45$
r_{11}	$rm \geqslant 6.94$ & $rm < 7.45$ & $lstat < 9.7$
r_{12}	$rm \geqslant 6.94$ & $rm < 7.45$ & $lstat \geqslant 9.7$

Table 2.1: Corresponding rules, defining the Decision Tree.

Every decision tree can be described by a set of decision rules which determnines its structure. A decision rule can be formally written through the subsets defining it. Let S_k denote the set of possible values of x and $s_{k,m}$ denote a specific subset $s_{k,m} \subseteq S_k$, then a decision rule with p conditions can be written as

$$r_m(\boldsymbol{x}) = \prod_{k=1}^{p} I(x \in s_{k,m}), \tag{2.11}$$

and $I(x)$ is the indicator function. A decision rule $r_m \in \{0,1\}$ is 1 if all of its p conditions are fulfilled and 0 otherwise. Let $\mathcal{T}$ denote a decision tree, $r_t, t = 1, \ldots, J$ its terminal node, and μ_t the value assigned to the t'th terminal node, then

$$\mathcal{T}(\boldsymbol{x}) = \sum_{t=1}^{J} r_t(\boldsymbol{x})\mu_t, \tag{2.12}$$

and each $r_t(\boldsymbol{x})$ consists of a set of conditions as in equation (2.11). In RuleFit not only the terminal nodes, but also the internal nodes are added to the ruleset. The reasoning is, that often an easier explanation is enough. We prefer easier explanations but if necessary the more complicated can be chosen by the model. Decision Trees have the tendency to overfit. Figure 2.1. shows a decision tree on the Boston Housing dataset, created with the rpart R-package. Table 2.1. shows the corresponding decision rules defining the Tree. Using equation (2.11) for example r_{11} can be expressed as

$$r_{11}(\boldsymbol{x}) = \prod_{k=1}^{3} I(x_k \in s_{k,11}) = I(rm \geqslant 6.94)I(rm < 7.45)I(lstat < 9.7).$$

Note that functions of form (2.11) can be easily incorporated into a regression model as dummy variables. In a way, the Decision tree can be seen as an exploratory step to find interesting subspaces, which can be integrated in a linear model as dummy variables to capture non-linear effects. Decision rules can not only be extracted from Decision Trees. There also exist clustering methods that produce results in the form of decision rules. However not all subspaces, or groups, help to explain the target variable. Found clusters might be totally unrelated with the goal of predicting y. In contrast the subspaces found by decision trees are formed with respect to y.

2.4.2 Combing the rules to an ensemble

Once a suitable set of decision rules is generated, they can be combined in a linear regression model of the form

$$y = \alpha_0 + \sum_{l=1}^{m} \alpha_l r_l(x) + \epsilon. \tag{2.13}$$

As $r_i(x) \in \{0, 1\}$ they already have the form of dummy variables and can be directly included in the regression model. A simple extension is to also include linear terms and combine them with the decision rules to the model

$$y = \alpha_0 + \sum_{j=1}^{p} \beta_j x_j + \sum_{l=1}^{k} \alpha_l r_l(x) + \epsilon. \tag{2.14}$$

This extension addresses the difficulty of rule and tree based methods to capture linear effects. The linear terms and decision rules complement each other, linear effects can be captured with linear terms and non-linear effects through decision rules. Splines, polynomials, time effects, spatial effects or random effects are straightforward extensions of equation (2.14).
Usually a large set of decision rules is necessary to have a high enough chance of finding good decision rules. This leads to a lot of unimportant and redundant decision rules. In terms of linear modeling, the ruleset induces a lot of noise covariates and multicolinearity. Additionally model (2.14) will always be high dimensional and often $p > n$. Regularization is a necessity in this situation.
RuleFit use the Lasso penalty [27]:

$$(\{\alpha_l\}_0^k, \{\beta_j\}_1^p) = \arg \min_{\{\alpha\}_0^k, \{\beta_j\}_1^p} \sum_{i}^{N} L\left(y_i, \alpha_0 + \sum_{j=1}^{p} \beta_k x_{ik} + \sum_{l=1}^{k} \alpha_l r_l(x)\right) + \tau \left(\sum_{l=1}^{k} |\alpha_k| + \sum_{j=1}^{p} |\beta_j|\right). \tag{2.15}$$

The Lasso penalty can be seen as a tradeoff between sparsity, measured in coefficient magnitudes, and the model fit on the training data. The Lasso is the most widely used form of regularization in machine learning literature and practice. It is easy to implement with very efficient solvers available. Secondly the Lasso sets many coefficient exactly to zero if the data is sparse or follows certain correlation structures. This is a desirable property as it allows easy interpretation of which variables are important and which not. And lastly argueably the Lasso enjoys its popularity by its formulation as an optimization problem. With that it fits more natural in the machine learning tradition as no probabilistic assumptions are necessary. On the other hand the Lasso can also be interpreted within the Bayesian framework. This allows to study the behaviour in a theoretical fashion and allows comparison to other regularization methods. One alternative is the horseshoe prior for regression. The following section compares the theoretical properties in (a) removing uninformative decision rules, (b) handling redundancy which leads to correlation between decision rules and (c) the solutions produced. It is argued here, that these three goals are better achieved by using the Horseshoe regression than by the Lasso.

2.5 Bayesian Regularization with the Horseshoe prior

The Horseshoe Hierarchy Recently Park & Casella [50] introduced the Bayesian Lasso. The Bayes Lasso is motivated by the observation that the Lasso can be interpreted as placing double exponential priors on the coefficients

$$p(\beta) = \prod_{j=1}^{p} \frac{\tau}{2} e^{-\tau|\beta_j|}, \tag{2.16}$$

and an independent prior $p(\sigma^2)$ on $\sigma^2 > 0$. [63]. The Lasso estimate then corresponds to the mode of the posterior distribution of β given the data [36]. Therefore in further comparison the bayesian Lasso and the horseshoe prior are compared if not stated otherwise. The double exponential distribution belongs to the family of scale mixture of normals. Another shrinkage prior in this family is the Horseshoe prior [15] [14]. The Horseshoe prior leads to the Bayesian Regression hierarchy

$$y|\mathbf{X}, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}_n(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n), \tag{2.17}$$
$$\beta_j|\lambda_j, \tau^2, \sigma^2 \sim \mathcal{N}(0, \lambda_j\tau^2\sigma^2), \tag{2.18}$$
$$\sigma^2 \sim \sigma^{-2}d\sigma^2, \tag{2.19}$$
$$\lambda_j \sim \mathcal{C}^+(0,1), \tag{2.20}$$
$$\tau \sim \mathcal{C}^+(0,1). \tag{2.21}$$

The main feature of the Horseshoe prior is that the shrinkage for β_j is determined by a local shrinkage parameter $\lambda_j > 0$ and a global shrinkage parameter $\tau > 0$. By that the Horseshoe prior can aggressively shrink noise through small values of τ, while allowing the signals to have large coefficients through large λ_j.

The global scale parameter Lasso and Horseshoe prior have the global shrinkage parameter τ in common, but they assume different prior distributions for τ, inverse-Gamma [1] and half-Cauchy respectively. One argument in favor of the half-Cauchy distribution is that it does not vanish at $\tau = 0$ [53]. This is in contrast to the inverse-Gamma which vanishes at $\tau = 0$. Therefore the Posterior density is biased away from $\tau = 0$ when the inverse-Gamma prior is used, as the prior assigns zero probability to this area. It has to expected that the half-Cauchy performs better especially in sparse situations, where the bias is the most impactful [53]. This is also the case for data in which very small magnitudes for $|\beta|$ are possible [30]. Another interesting feature of the half-Cauchy distribution is its heavy tails allowing the data to dominate through the likelihood, dispite our prior specification [30] and is therefore weakly informative. In emprical comparisons the horseshoe prior leads to smaller values of τ, when the data is sparse allowing a more aggressive shrinkage of noise covariates [15].

The local scale parameter The horseshoe prior is a global-local shrinkage prior. The introduction of a separate individual shrinkage parameter leads to interesting properties. Most importantly when τ is very small, individual β_j can still be large through large λ_j. The heavy tails of the half-Cauchy distribution on λ_j allow truly informative predictors to escape the gravitation towards zero. The full conditional posterior distribution of λ_j depends on both τ and β_j. In situations when τ is small the model has to counteract with large λ_j, if a predictor is informative. Therefore when both the true β_j is large and τ is small the λ_j has to become extremly large. The half-Cauchy prior distribution allows this and the horseshoe can therefore adapt to a variety of different pattern of sparsity and effectsize.
Under the double-Exponential prior (Lasso) τ must balance between the risk of under-shrinking the noise and the risk of over-shrinking large signals [15]. For small τ also

[1] The double Exponential is a special case of the inverse-Gamma distribution.

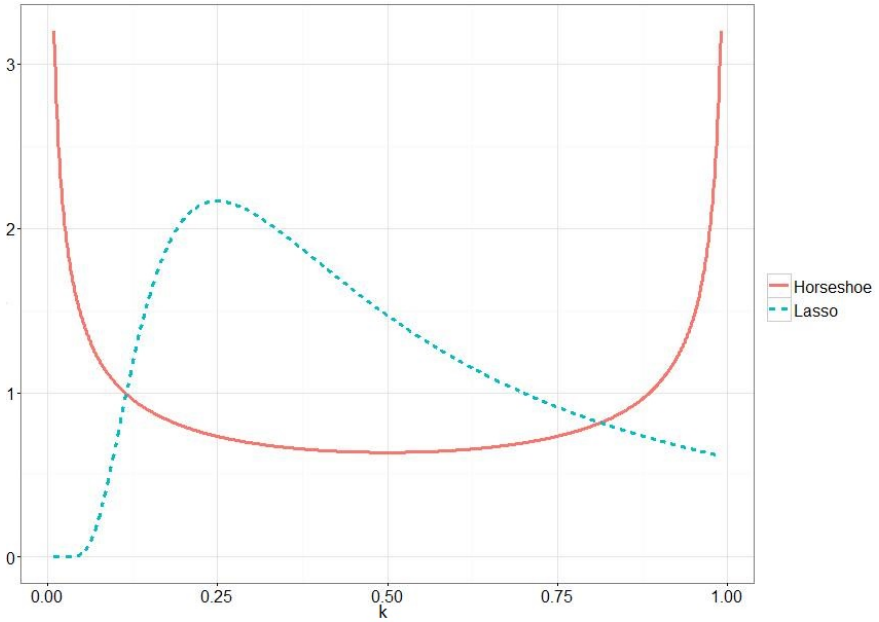


Figure 2.2: The prior distribution of the shrinkage coefficient κ under the Horseshoe prior and the Lasso (Double-Exponential). $\kappa = 0$ implies no shrinkage, $\kappa = 1$ total shrinkage.

informative predictors will inevitable be shrunken under the Lasso. In practice τ is usually determined via cross validation, but the Lasso solution will be a compromise between these two competing risks. If the level of noise is high, the double-Exponential will shrink the informative predictors as well. This is problematic both in terms of interpretation and prediction risk.

Shrinkage behaviour Shrinkage priors can be studied through the shrinkage coefficients defined as $\kappa_j = \frac{1}{1+\lambda_j^2}$. $\kappa_j = 0$ implies total shrinkage and $\kappa_j = 1$ no shrinkage on β_j. This comes from the observation that if assuming fixed $\tau = \sigma^2 = 1$

$$E(\beta_j|y) = \int_0^1 (1-\kappa_j) y_i p(\kappa_j|y), \tag{2.22}$$

$$= \{1 - E(\kappa_j|y)\} y_j. \tag{2.23}$$

The horsehoe prior implies $\kappa_j \sim Beta(0.5, 0.5)$, while the double-Exponential prior implies $p(\kappa_j) = k_j^{-2} e^{-\frac{1}{2\kappa_j}}$ [15]. Figure (2.2) shows the distribution of the κ_j under horseshoe and double exponential prior. The shrinkage profile exhibits important differences. The double-Exponential vanishes at $\kappa_j = 0$, which means that important predictors can not be left unshrunk. Most probability mass is on small to medium values of κ_i and considerably high mass on $\kappa_i = 0$. This allows to shrink unimportant predictors to zero.
In contrast the shrinkage implied by the horseshoe prior is unbounded at both $\kappa = 0$ and $\kappa = 1$, giving it the U-shape. The unboundness at 0 allows predictors to be left unshrunk and the unboundness at 1 produces a heavy pull of unimportant predictors towards zero. Therefore when using the Horseshoe prior we expect a priori that predictors are either noise or signal, that is: either removed from the model or left unshrunk. The shrinkage profile of the double-Exponential prior with its high mass on intermediate values of κ_i is not that clear. One interpretation might be, that we do not trust large effects and conservativly prefer to

shrink large coefficients. In the context of Rule Ensemble learning the shrinkage profile of the Horseshoe prior seems more appealing. As we seek to get a compact set of decision rules, we would like to keep truly informative rules unshrunk and all other rules removed from the set. Intermediate behaviour, which keeps more or less informative rules leads to a lower interpretability.

Relationship to Spike and Slab The most commonly used method for variable selection in the bayesian community is the Spike and Slab model where a discrete-mixture prior

$$\beta_j \sim w \cdot g(\beta_j) + (1 - w) \cdot \delta_0, \tag{2.24}$$

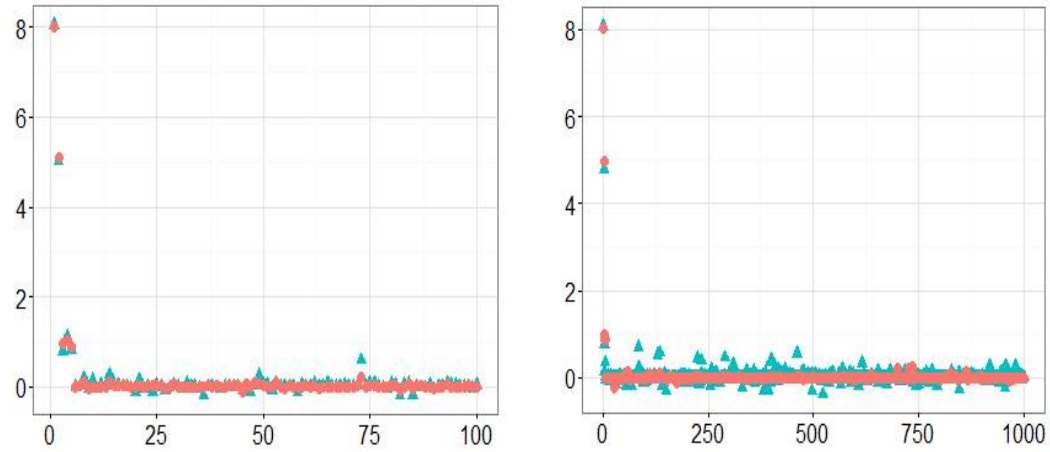
is asssumed and w denotes the inclusions probability of predictor x_j [31]. The Spike and Slab model always takes only a number of predictors in the model and goes through different combinations of predictors while sampling. (2.24) is often refered to as two-group model, one group of predictors that are included and one group of the predictors that are excluded. Polson & Scott [51] show that the Horseshoe prior can be seen as an approximation of the model (2.24), with τ as the analogous to w and $g(\cdot)$ being the Cauchy distribution. Bhattacharya et al.[9] report that the marginal posterior distributions of β under the horseshoe are similar to the result one would get by using the discrete-mixture model. However the computations under the discrete-mixture model are much more demanding as it needs to explore a modelspace of size 2^p. Another advantage of the Horseshoe prior is its adaptivity. The prior is only weakly informative, in contrast to Spike and Slab models, which are quite sensitive to the hyperparameters. This property seems especially important in this work. The rule ensemble will look differently in each run, varying in number of rules, number of uninformative rules and covariance structure, even when run on the same data set. Specifying hyperparameters in this situation is a Sisyphos task.

Handling noise The above stated theoretical differences result in different empirical behaviour in situations where $\boldsymbol{X}$ contains uninformative predictors. Generally the Horseshoe prior performs better than the Lasso, when noise is present. The Lasso leads to overshrinkage of large β due to the direct conflict between squelching the noise towards zero and not shrinking important predictors [53]. The difference increases with the number of uninformative predictors, but depends on the structure of β as well. Especially the magnitude of the non-zero predictors plays an important role. When the signal is large, it can lead to an undershrinking of the noise components.

Simulation 1 This simulation illustrates the difference between Lasso and Horseshoe regression when it comes to handling noise. Let

$$X_1, \ldots, X_p \sim \mathcal{N}(0, 1), \tag{2.25}$$
$$y = 8 \cdot X_1 + 5 \cdot X_2 + X_3 + X_4 + X_5 + \epsilon, \tag{2.26}$$
$$\epsilon \sim \mathcal{N}(0, 1). \tag{2.27}$$

Only the first 5 predictors effect y while the other predictors are noise. Figure 2.3 shows the estimated coefficients under $p = 100$ and $p = 1000$ and $n = 100$. The Lasso estimation is taken as the optimal under cross-validation and for the Horseshoe prior the posterior mean is taken. With $p = 100$ the results are similar, but already the Horseshoe is a bit better in squelching the noise towards zero. The difference becomes sincere with $p = 1000$. For the Horseshoe there seems to be no difference, the noise components are still shrunk towards zero. For the Lasso on the other hand the difference is quite visible, as many coefficients escape zero. Even more sincere some of the noise components reach a magnitude close to 1. This means that some of the true signals are no longer distinguishable from the noise. The



(a) Coefficients of Lasso (green) and Horseshoe (orange) with p=100.

(b) Coefficients of Lasso (green) and Horseshoe (orange) with p=1000.

Figure 2.3: Difference between Horseshoe prior and Lasso in shrinking uninformative covariates.

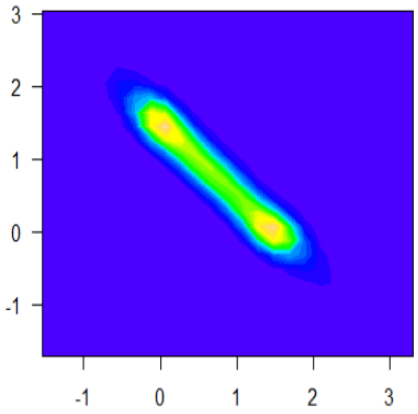
deterioration of the Lasso can be easily understood from the theoretical properties above. As some of the coefficients are quite large τ can not be chosen to be very small, which would be necessary to shrink the noise efficiently. With increasing p smaller values of τ are needed which is in direct conflict with keeping the true β large.

Handling correlation Another problem with the lasso is that its optimality in finding the right predictors is only guaranteed under the restrictive condition, that the correlation between predictors is under a certain degree. This problem is well studied as the *irrepresentability condition* [65]. The Lasso fails to select the right predictors, when the correlation between important and unimportant predictors is too high. This stems from the previously described over-shrinkage of the important predictors: The Lasso takes correlated predictors to substitute for the overshrinkage. The irrepresentabily condition will be violated in most cases, as the correlation between the decision rules is high.
On the other hand the horseshoe prior was shown to be relatively robust against correlated predictors in terms of model constistency but even more so in terms of predictive performance [64]. This can be understood from the above noted similarity to discrete-mixture models. However most literature on global-local shrinkage priors focusses on sparsity with noisy components and asssumes orthogonal Design. The behaviour in situations with non-orthogonal $\boldsymbol{X}$ is understudied [9].

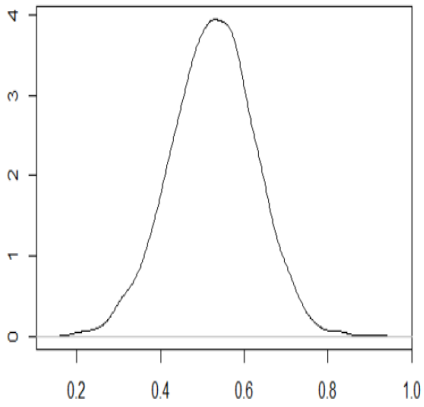
Simulation2 To illustrate the difference in handling correlated predictors let

$$\begin{aligned} x1, x2 &\sim \mathcal{N}(0,1), \\ x3 &\sim \mathcal{N}(x1, 0.15), \\ y &\sim \mathcal{N}(x1 + x2, 0.5), \end{aligned}$$

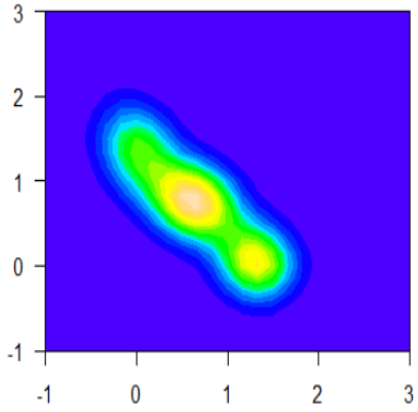
with $n = 100$. The predictors x1 and x3 are highly correlated ($\rho = 0.99$) and x1 and x2 influence y. Figure 2.4 shows the posterior distributions for β_1, β_2 and β_3 under the Horseshoe and under the Lasso. For the Lasso the β values obtained within the 95% quantile of τ are considered. The reason is that in practice for the Lasso one value for τ is chosen and the cross validated τ value is very likely to lay in this region. The joint posterior of β_1 and β_3 is clearly bimodal under the horseshoe prior, with values in between due to the Gibbs Sampling. Note that for each of the modes one of the predictors is exactly zero and the other has the true value



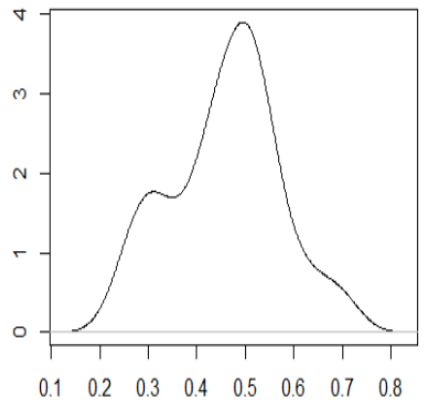
(a) Joint-posterior of β_1 and β_3 under the Horseshoe



(b) Posterior distribution β_2 under Horseshoe



(c) Joint-posterior of β_1 and β_3 under the Lasso



(d) Posterior distribution β_2 under the Lasso

Figure 2.4: Posterior densities. The Horseshoe posterior leads to bimodal distributions, with either x1 or x3 equal to zero. The Lasso more unimodal with mode in between the horseshoe mode. Under Horseshoe uncorrelated predictor x2 not effected by multicollinearity.

1.5. In a way the horseshoe produces two different models. In each one of the correlated predictors is excluded. The posterior distribution of β_2 is uneffected by the multicolinearity.
On the other hand the joint posterior under the Lasso has a mode in between the two horseshoe modes around $(0.9, 0.6)$. The Lasso has to find a compromise: clearly not both predictors can be included with full magnitude. However with higher global shrinkage there is no individual shrinkage parameter which allows one of the β_1 or β_3 to become large and by that set the other coefficient to zero. Instead the found solution shrinks both coefficients equally. Also under the Lasso the posterior distribution of β_2 is slightly distorted from the overall shrinkage.
Under Correlation the Horseshoe prior leads to multimodal distributions, one mode for each correlated predictor. Each of these modes reflects a model, where one predictor is chosen, with coefficient close to the true one, while the other coefficients are set to zero. Therefore taking the posterior mean mimics bayesian model averaging over the models using different predictor combinations. Uncorrelated predictors are generally uneffected under the horseshoe, as they can escape the global shrinkage through their local shrinkage parameter.
The Lasso on the other hand seeks to find a compromise and leads to a unimodal posterior distribution between the true values. Also multicollinearity between predictors induces a higher level of global shrinkage, which can distort the posterior distributions of uncorrelated predictors as well.

Multimodality It was pointed out in the previous paragraphs that the Horseshoe prior can lead to multimodal posterior distributions of $\boldsymbol{\beta}$. Predictors often will have one spike at zero which represents their probability of being excluded from the model. If the predictor is informative it will have a second spike at a non-zero value. Also if there is correlation between predictors it will lead to multimodality, as discussed previously. This aspect of the Horseshoe prior is understudied. Most studies focus on sparse or ultra sparse situations, where multimodality is unlikely to occur, as in this situation predictors are either clearly noise or signal.
It needs further clarification, but in my opinion it is exactly this multimodality which leads to the similarity to the Spike and Slab models. Each mode can be interpreted as a model one would obtain using discrete-mixture priors, with a number of predictors set to zero and a number of non-zero predictors. Carvalho et al.[15] tackle this phenomenon by defining

$$\omega_j = \mathbb{E}\Big(\frac{1}{1+\lambda_j^2}\Big) \tag{2.28}$$

as pseudo-inclusion probability of a predictor β_j in analogy to the inclusion probabilites in discrete-mixture models. In discrete-mixture models the predictive performance depends on the ability of the algorithm to explore the interesting regions of the modelspace. This implies that, when using the Horseshoe prior, good predictive performance crucially depends on the ability of the algorithm to explore the different modes [42]. The modelspace for discrete-mixture models is 2^p so it has to be expected that this is also the number of possible modes in the posterior distribution of $\boldsymbol{\beta}$. In practice that means, that it if posterior samples are obtained via Markov Chain Monte Carlo (MCMC), the behaviour of the chain should be checked. Slow mixing behaviour makes it unlikely for the model to visit all the different modes, if not a high enough number of iterations is chosen.

Horseshoe+ An extension to the Horseshoe hierarchy is to introduce a further mixing variable

$$\lambda|\eta \sim \mathcal{C}^+(0,\eta), \tag{2.29}$$
$$\eta \sim \mathcal{C}^+(0,A), \tag{2.30}$$

with $A = 1$ being the Horseshoe+ estimator [7]. The Horseshoe+ estimator pulls noise even stronger towards zero and signals away from zero. Other choices of A will be discussed in Section 3.2.

3 Method

This chapter explains the details for the implementation of the Horseshoe RuleFit. It follows the same structure as the previous chapter, starting with the used Ensemble of Tree method, over to the rule extraction and considerations which rules to take, to the Gibbs Sampling and further considerations about Hyperpriors and finally methods which allow easy interpretable summary of the obtained rule ensemble. The reader will notice that at many points decisions are made, for example which ensemble of tree method to choose. Decisions are always evaluated by the three criteria to (1) get an as compact and understandable ruleset as possible, while (2) achieve high predictive accuracy and (3) keep the model computationally feasible. As to my knowledge there has not been any Bayesian approaches to RuleFit I expect the here presented method to be a starting point, with many improvements possible.

3.1 Implementation of the Rule Generating Process

3.1.1 Choice of Ensemble method

Friedman & Popescu [27] choose gradient boosting, as the best performing ISLE algorithm in their previous experiments [26]. Most tree based rule ensemble methods follow this reasoning. However this choice might not be optimal. In gradient boosting, each tree is fit on the pseudo residuals of the current ensemble [25]. This means that the produced trees and decision rules are dependent on the rest of the ensemble. They try to correct, what the previous ensemble misclassified. It might not be possible to set a lot of this decision rules to zero without destroying this dependency structure. The produced rules might often not be individually informative but only when combined to an ensemble. In the rule ensemble we would like to get a compact set of decision rules, which explain most of the variance, by setting a vast amount of coefficients to zero. This means that, even though gradient boosting works great as an ensemble method, it does not imply that it is also the best choice for generating decision rules.
In Random Forest on the other hand each tree is independent from all previous trees. Each tree tries to find the individual best partitioning, given a random subset of observations and covariates. The rules are therefore less dependent on each other. Random Forest will produce more redundant, uninformative rules compared to gradient boosting, as it will often choose very similar splits, and due to the covariate sampling is often forced to use uninformative

predictors. At the same time, the good rules found in Random Forest will be individually good and not dependent on other rules. The horseshoe prior is well suited to deal with redundancy and with uninformative decision rules. Therefore Random Forest appears to be a good choice for generating decision rules as well.
A second argument in favour of Random Forest is, that it was shown to be overall the most reliable and stable classifier. Fernandez et al. [22] found in an extensive study on 121 real world data sets that the most reliable classifiers when run with standard settings are Random Forest and SVM.
As both Random Forest and Boosting seem to have data sets in which they work well and they produce different kind of rules a compromise is to use rules generated from both Gradient Boosting and Random Forest. This makes the result less dependent on our choice. Also the bigger variety of rules, makes the algorithm more adaptive to different data sets as it increases the probability of finding good rules.

3.1.2 Rule Generation Hyperparameters

The shape of the rules obtained from the ensemble of trees is determind by the ensemble method but as well by the parameters used. This aspect is not very explored in the literature. Usually of interest is how the parameter choices effect the predictive accuracy of the whole tree ensemble, where the shapes of the actual rules is only of minor interest, as they are hardly interpretable anyways. It was found in several studies that especially the number of trees used, the number of variables used in each split and the shrinkage ν and tree depth for the gradient boosting have the largest impact on the ensembles accuracy. The few studies that investigate the effect of parameters on rule ensembles for example [46] focus on the same parameters.
These results are certainly important, but it might be a mistake to focus on those exclusively. Random Forest grows each tree to full depth, which useually is a severe overfitting. By that the variance between the trees increases, which then decreases the bias. This ensemble effect is well studied. The great predictive performance comes from averaging over all those individually overfitted trees. But what happens if we decompose the ensemble and remove most of them is not obvious. It is to be expected that the balancing effect that make Random Forest and Boosting so stable disappears and overfitting might occur. Dembczynski et al. [21] argue that in order to use tree generated rules more parameter tuning needs to be performed.
Friedman & Popescu [27] identified the maximum depth of the trees as important parameter, as it determines the complexity of the produced rules. Choosing the maximum depth too high will lead to very specific rules, which will result in overfitting. On the other hand, if the relationship is highly non-linear but smooth very complex rules are necessary to approximate this relationship. They recommend to chose the tree depth indirectly by defining the number of terminal nodes for each tree as,

$$t_m = 2 + fl(\psi) \tag{3.1}$$

and drawing ψ from a exponential distribution with probability density function

$$p(\psi) = \frac{1}{L-2} exp(\frac{-\psi}{L-2}), \tag{3.2}$$

where $fl(x)$ is the largest integer less or equal than x [27] . $L \geqslant 2$ controls the *average* number of terminal nodes. Setting $L = 2$ will produce only tree stumps consisting of one split. With this indirect specification the forest is composed of trees of varying depth, which allows the forest to be more adaptive to the data and makes the choice of a suitable tree depth less important.
Another parameter to consider is the minimum number of observations in the terminal nodes. Random Forest typically uses 1 for classification and 5 for regression, as it tries to achieve

overfitting in the individual trees. Hence to avoid too specific rules the number of observations should not be chosen too small. Afterwards the rules with too low support can be removed, but better rules might be found if the search in the tree focuses on actually usable decision rules. In my experiments I found a minimum nodesize of 10 for regression and 6 for classification to work well.
Friedman & Popescu [27] and Meza [46] state that the sample size η used for each tree does not have a big effect, as long is it is chosen low enough $\eta \leqslant \frac{N}{2}$ and $\eta \sim \sqrt{N}$ as N becomes large. This setting is also used in this work.

3.1.3 Rule Correlation

One challenge is the high correlation between the decision rules. Decision rules extracted from tree ensembles inherit their correlation structure. That is:

1. The correlation between siblings in the first split is $\rho = -1$.
2. The correlation between parent node and child node is $|\rho| \in (0, 1)$.
3. Depending on the specific tree ensemble method the correlation between trees is usually moderately high. There always is a lot of redundancy over the forest. Especially the very informative variables are often used for splits, leading to highly correlated decision rules.
4. Each parent node together with it's childnodes are perfectly colinear.

Highly correlated features can pose big problems in regression models, even more so in the $p >> n$ situation. The horseshoe regression was shown to be relatively robust against correlated features, however in this extreme situation problems remain. If perfectly correlated features are left in the rule ensemble the estimation gets unstable and extreme behavior can occur. Also there is no information gain by keeping perfectly correlated features, as it is impossible for the model to decide which variable to keep.
To prevent this pathologies I decided to remove perfectly correlated predictors. If variables are perfectly correlated it is only necessary to keep one of them. Additionally I chose to remove rules $r_l(x), r_j(x)$ if their correlation ρ exceeds a threshold $\rho_{l,j} > c$. As the variables are slightly different it makes sense to keep the rule with the lowest error. The procedure is done with the following steps:

1. For each decision rule $r_l(x), l = 1, ..., m$ calculate the correlation with the other decision rules $Cor(r_l(x), r_j(x)) l \neq j$
2. If the correlation is higher than the threshold c, keep only the decision rule with lowest error.

As a standard setting I found $c = 0.99$ to work well, as it removes just enough rules, so that the estimation remains stable. Also for each split I chose randomly only one of the two sibling decision rules to include and drop the other. Doing that avoids the perfect multicollinearity and also simplifies calculations greatly as only half of the rules are kept.

3.1.4 Rule Pruning

Another problem in decision rules can be best described as overcomplication. Consider a decision rule where only the first condition is informative, while the second split does not reduce the error significantly. This happens for example in Random Forest when a tree is forced in a split to use an uninformative variable due to the covariate sampling. In terms of interpretation this is very undesirable, as it suggests an interaction where in fact is most likely none. It also might lead to overfitting. To prevent this behavior I decided to apply Rule

Pruning. Each rule $r_l, l = 1, ..., m$ gets pruned if the change in error of this rule is lower than a threshold t when using the pruned rule instead. Conditions are removed, if it does not make a difference. This is desirable for both interpretability and better generalization. As threshold I found that removing conditions, if the relative error increases less than 0.01 works well.

3.1.5 Rule Support

Also to consider is if it is useful to include Decision rules with very low support. In an extreme it can happen that a rule consists only of one observation. This problem is especially to consider when using the Horseshoe regression. The Lasso shrinks coefficients globally, so that this rules most likely will get removed. The Horseshoe prior however allows the signal to escape this global shrinkage and this single observation rules are most certainly signal, as they help to explain the training data (perfectly). This might lead to overfitting and also lower interpretablity. To prevent this I decided to remove all rules with a support $s(r_l) = \frac{1}{N}\sum_{i=1}^{N} r_l(x)$ lower than 0.01 or higher than 0.99. If prior information about the size of potential groups in the data is available this setting can be adjusted.

3.1.6 Implementation in R

For creating the trees the R-packages randomForest and gbm were used and extended with own code to allow the sampling of the tree depth described in 3.1.2. Extracting the rules was done with the very convenient R-package inTrees which allows to decompose Forests into decision rules.

3.2 Implementation of the Horseshoe Regression

3.2.1 Gibbs Sampling

Sampling from the standard horseshoe hierarchy (2.17) is not straightforward, as the conditionals for the hyperparameters λ_j and τ do not follow standard distributions. Slice sampling can be applied, as described in [48] and implemented in the R-package monomvrm, but is computationally expensive.
This issue is of big importance as in RuleFit due to the induction of a large number of decision rules $\boldsymbol{X}$ is always high dimensional. Computational efficiency is of special importance in this context.
Makalic and Schmidt [44] propose a revised Horseshoe hierarchy that exploits the representation of a half-Cauchy distributed random variable $x \sim \mathcal{C}^+(0, A)$ through

$$x^2|a \sim \mathcal{IG}\Big(\frac{1}{2}, \frac{1}{a}\Big), \tag{3.3}$$

$$a \sim \mathcal{IG}\Big(\frac{1}{2}, \frac{1}{A^2}\Big) \tag{3.4}$$

which leads to conjugate conditional posterior distributions. With introduction of the auxiliary variables $\nu_1, ..., \nu_p$ and ψ the horseshoe hierarchy becomes

$$y|\mathbf{X}, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}_n(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n), \tag{3.5}$$
$$\beta_j|\lambda_j, \tau^2, \sigma^2 \sim \mathcal{N}(0, \lambda_j\tau^2\sigma^2), \tag{3.6}$$
$$\sigma^2 \sim \sigma^{-2}d\sigma^2, \tag{3.7}$$
$$\lambda_j{}^2|\nu_j \sim \mathcal{IG}\Big(\frac{1}{2}, \frac{1}{\nu_j}\Big), \tag{3.8}$$
$$\tau^2|\psi \sim \mathcal{IG}\Big(\frac{1}{2}, \frac{1}{\psi}\Big), \tag{3.9}$$
$$\nu_1, \ldots, \nu_p, \psi \sim \mathcal{IG}\Big(\frac{1}{2}, 1\Big). \tag{3.10}$$

In their paper [44] they also present the full conditional distributions, required for Gibbs sampling. The sampling scheme is to first sample $\boldsymbol{\beta} \in \mathbb{R}^p$ with

$$\boldsymbol{\beta}|\sigma^2, \lambda_1{}^2, \ldots, \lambda_p{}^2, \tau^2 \sim \mathcal{N}_p(\mathbf{A}^{-1}\mathbf{X}^T\mathbf{y}, \sigma^2\mathbf{A}^{-1}) \tag{3.11}$$

where

$$\begin{aligned}\mathbf{A} &= (\mathbf{X}^T\mathbf{X} + \boldsymbol{\Lambda}_*{}^{-1}), \\ \boldsymbol{\Lambda}_* &= \tau^2\boldsymbol{\Lambda}, \\ \boldsymbol{\Lambda} &= \mathrm{diag}(\lambda_1{}^2, \ldots, \lambda_p{}^2).\end{aligned} \tag{3.12}$$

The full conditional for σ^2 follows a Inverse-Gamma distribution of the form

$$\sigma^2|\boldsymbol{\beta}, \lambda_1{}^2, \ldots, \lambda_p{}^2, \tau^2 \sim \mathcal{IG}\Big(\frac{n+p}{2}, \frac{\epsilon^2}{2} + \frac{\boldsymbol{\beta}^T\boldsymbol{\Lambda}_*{}^{-1}\boldsymbol{\beta}}{2}\Big) \tag{3.13}$$

where $\epsilon^2 \in \mathbb{R}$ is the sum of squared errors (SSE) $\epsilon^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$. After sampling $\boldsymbol{\beta}$ and σ^2 the shrinkage parameters can be sampled from

$$\lambda_j^2|\nu_j, \beta_j, \tau^2, \sigma^2 \sim \mathcal{IG}\Big(1, \frac{1}{\nu_j} + \frac{\beta_j{}^2}{2\tau^2\sigma^2}\Big) \quad , j = 1, \ldots, p \tag{3.14}$$

$$\tau^2|\psi, \boldsymbol{\beta}, \lambda_1{}^2, \ldots, \lambda_p{}^2, \sigma^2 \sim \mathcal{IG}\Big(\frac{p+1}{2}, \frac{1}{\psi} + \frac{1}{2\sigma^2}\sum_{j=1}^{p}\frac{\beta_j{}^2}{\lambda_j{}^2}\Big) \tag{3.15}$$

and the auxiliary variables from

$$\nu_j|\lambda_j^2 \sim \mathcal{IG}\Big(1, 1 + \frac{1}{\lambda_j^2}\Big), \tag{3.16}$$
$$\psi|\tau^2 \sim \mathcal{IG}\Big(1, 1 + \frac{1}{\tau^2}\Big). \tag{3.17}$$

Note that all distributions are easy to sample from and efficient sampling procedures are implemented in standard statistical software. However the step (3.11) requires taking the inverse of a $p \times p$ matrix which is of complexity $\mathcal{O}(p^3)$. This inverse can not be precomputed as the Λ_* changes in every iteration. Since through the induction of a large number of decision rules p allways will be large this step can be computationally demanding.Bhattacharya et al. [8] propose an alternative exact sampling algorithm for $\boldsymbol{\beta}$ in global-local prior regression models such as the horseshoe regression. Suppose we want to sample

$$\beta \sim \mathcal{N}(\mu, \Sigma), \quad \Sigma = (\Phi^T\Phi + D^{-1})^{-1}, \quad \mu = \Sigma\Phi^T\alpha. \tag{3.18}$$

Algorithm 2 Bhattacharya, Chakraborty and Mallick (2015)

1: Set $\Phi = \frac{1}{\sigma}\boldsymbol{X}$, $D = \sigma^2\boldsymbol{\Lambda}_*$ and $\alpha = \frac{y}{\sigma}$
2: Sample $u \sim \mathcal{N}(\boldsymbol{0}, D)$ and $\delta \sim \mathcal{N}(\boldsymbol{0}, I_n)$
3: Set $v = \Phi u + \delta$
4: Solve $(\Phi D \Phi^T + I_n)w = (\alpha - v)$
5: Set $\beta = u + D\Phi^T w$

Instead of sampling from (3.11) directly, we can solve a series of linear systems, see Algorithm 2.

This indirect sampling can be carried out with $\mathcal{O}(n^2 p)$ operations and therefore scales especially well for $p >> n$. It is used in this work when $p > n$. The details and a proof can be found in the original paper. For the case of $n > p$ the Rue algorithm for Sampling from Markov Random Fields [56] can be applied [8], as shown in Algorithm 3.

Algorithm 3 Rue (2001)

1: Compute the Cholesky decomposition $\Phi^T\Phi + D^{-1} = LL^T$.
2: Solve $Lv = \Phi^T\alpha$.
3: Solve $L^T m = v$.
4: Sample $z \sim \mathcal{N}(0, I_p)$.
5: Solve $L^T w = z$.
6: Set $\beta = m + w$.

In any case taking the inverse directly for sampling should be avoided, as numerical instabilities can occur. This is especially the case, as the values in $\boldsymbol{\Lambda}$ can get very high, when the number of unimportant predictors increasing. Rues algorithm requires computing the Cholesky decomposition therefore if possible Algorithm 2 is to prefer, when n and p are similar.

3.2.2 Classification

In a Bayesian framework Classification is usually either done via the addition of a Metropolis-Hastings step in the Gibbs sampler, or via data augmentation. The data augmentation approach has the clear advantage that it is computationally relatively inexpensive and can be easily integrated in the Gibbs Sampling Scheme. The first and most commonly used data augmentation is the Albert and Chibs (AC) algorithm for probit regression [4].
In practice the usage of the AC algorithm is problematic, as it leads to very slow convergence. Additionaly the AC algorithm is only assured to converge if $n > p$ and $\boldsymbol{X}$ has full rank. Chakraborty & Khare [16] recently proposed a novel data augmentation approach for probit regression which leads to better convergence even when $p >> n$ (Algorithm 4).

Algorithm 4 Haar PX-DA by Chakraborty and Khare (2016)

1: Sample independent $z_i \sim \mathcal{TN}(\boldsymbol{x_i}^T\boldsymbol{\beta}, 1, y_i)$, $i = 1, \ldots, N$
2: Calculate $\mathcal{A}(\boldsymbol{z}) = \boldsymbol{z}^T X(I_n - (X^T X + {\boldsymbol{\Lambda}_*}^{-1})^{-1}X^T)\boldsymbol{z}$
3: Sample $u \sim Gamma\left(\frac{n}{2}, \frac{2}{\mathcal{A}((z)}\right)$
4: Set $g = \sqrt{u}$
5: Set $\boldsymbol{y}_* = g\boldsymbol{z}$

Note that step 2 requires taking the Inverse, which can be daunting for big p, as previously discussed. However the improvement in mixing is substantial [16]. For sampling β

Algorithm 3 can be applied, as the Cholesky already was computed, therefore the loss in computational efficiency is only significant for $p >> n$ situations.
A recently proposed alternative to the bayesian probit regression is the polya-gamma data augmentation approach for bayesian logistic regression[52]. They extend the latent variable idea of Albert and Chibs to binary regression with log-link and show that it has good mixing properties. As the probit data augmentation it is easily implemented in a Gibbs sampling procedure. Let PG denote the Poly-Gamma distribution then $X \sim PG(b, c)$ has the probability density function

$$f(x) = \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{g_k}{(k-1/2)^2 + c^2/(4\pi^2)} \tag{3.19}$$

with $g_k \sim Ga(b, 1)$. Polson et al. [52] implemented an efficient algorithm for sampling from the distribution (3.19) in their R-package BayesLogit. Posterior Sampling for binary regression is done with the two-step procedure

$$w_i|\boldsymbol{\beta} \sim PG(1, x_i^T\boldsymbol{\beta}) \tag{3.20}$$
$$\boldsymbol{\beta}|y, \boldsymbol{w} \sim \mathcal{N}(m_w, V_w) \tag{3.21}$$

with

$$V_w = (X^T\Omega X + B^{-1})^{-1} \tag{3.22}$$
$$m_w = V_w(X^T\kappa) \tag{3.23}$$
$$\kappa = \left(y1 - \frac{1}{2}, \ldots, y_n - \frac{1}{2}\right)$$
$$\Omega = diag(w_1, \ldots, w_n)$$

Note that (3.22) differs from the standard sampling scheme for the Horseshoe regression. The efficient sampling algorithms from the previous section cannot be used.

3.2.3 Separation

One problem that can occur specifically in binary classification is in the literature referred to as separation. Albert & Anderson [3] make the distinction between complete separation and quasi complete separation. Formally complete separation occurs if there exists a vector $\alpha = (\alpha_1, \ldots, \alpha_p)$ that for every $i = 1, \ldots, n$,

$$x_i^T\boldsymbol{\alpha} < 0 \;\; if \;\; y_i = 0 \quad , \; x_i^T\boldsymbol{\alpha} > 0 \;\; if \;\; y_i = 1, \tag{3.24}$$

and quasi complete if for every $i = 1, \ldots, n$,

$$x_i^T\boldsymbol{\alpha} \leqslant 0 \;\; if \;\; y_i = 0 \quad , \; x_i^T\boldsymbol{\alpha} \geqslant 0 \;\; if \;\; y_i = 1. \tag{3.25}$$

If complete separation in one covariate x_j occurs the MLE estimate of its coefficient β_j will go to $-\infty$ or $+\infty$ [66]. This is intuitively understandable: if there is no negative examples $y = 0$ for $x_i^T\boldsymbol{\alpha} > 0$ *limiting* the log-odds, then the chance of $y = 1$ is infinity higher for $x_i^T\boldsymbol{\alpha} > 0$ and vice versa. One can think of the table

	y=0	y=1
$x_i^T\boldsymbol{\alpha} \geqslant 0$	$a_{1,1}$	$a_{1,2}$
$x_i^T\boldsymbol{\alpha} \leqslant 0$	$a_{2,1}$	$a_{2,2}$

If any of the cells is empty the quasicomplete condition is fulfilled. If both $a_{1,1}$ and $a_{2,2}$ are

empty complete separation occurs. A covariate x_j is a solitary seperator [32] if separation occurs in this variable according to the above definitions and

$$\alpha_j \neq 0 \ , \alpha_r = 0 \quad \forall \, j \neq r \tag{3.26}$$

which implies that for $\alpha_j > 0$ for all $i = 1, \ldots, n$

$$x_{i,j} \geqslant 0 \ \ \forall \, y_i = 1 \ \ , x_{i,j} \leqslant 0 \quad \forall \, y_i = 0. \tag{3.27}$$

Intuitively a solitary separator means, that the data is perfectly separated with a line at $x_j = 0$, while in the case of complete seperation an intercept is necessary in any case to separate the data perfectly. Ghosh et al. [32] show that under Cauchy priors a posterior mean for $\boldsymbol{\beta}$ exists only in the absence of solitary separators. In empirical studies [32] also show that in presence of complete or quasi-complete separation the Cauchy prior leads to extreme behavior, namely unreasonable high posterior means and sample auto correlation to an degree of unusability. This stems from the fat tails of the Cauchy distribution, which allows the β to become very large. On one hand, this property of the Horseshoe prior is very attractive for identifying truly informative predictors, but in the context of classification it bares a risk and special attention should be given to the phenomena of separation. This is especially true in the context of rule ensembles. Decision trees *try* to separate the data perfectly. Therefore when combining the resulting decision rules to the ensemble it is likely to have complete separation, especially when the data is easy to separate and if $p >> n$. To see this imagine a tree leaf which contains only $y = 1$ observations. This leads to the table As a_{11} is empty, the quasicomplete condition

	y=0	y=1
$r_l(x)^T = 0$	0	$a_{1,2}$
$r_l(x)^T = 10$	$a_{2,1}$	$a_{2,2}$

is fulfilled. Also note that if a decision rule separates the data perfectly it will be a solitary separator, as no intercept is necessary to separate at $r_l(x) = 0$. It therefore seems like a good idea, to center the decision rules as well, as then $r_l(x)$ is no longer a solitary separator. Ghosh et. al [32] show that after centering of dummy variables the corresponding MCMC-chains for β do not diverge any more, but they are still effected by the separation. This means that the Classification is not a straightforward extension, but the separation problem must be addressed.

There has been different approaches to adress this problem, however most of them being concerned with the maximum likelihood estimate. Another often used approach is to simply drop the covariates leading to perfect separation [66]. This approach seems unsatisfying as those covariates are potentially the most informative ones. In the Bayesian framework one way is to use lighter tailed priors, example given t-student priors or normal priors instead, which show less extreme behaviour in presence of seperation [32].

Another Issue is that if $p > n$ the data always is quasi-complete separated, leading to poor mixing under the Cauchy prior [32]. This follows from the linear dependency of $\boldsymbol{X}$, which always allows to find $\boldsymbol{\alpha}$ such that the quasi complete separated condition holds with equality for all observations. As a consequence the number of decision rules induced should not exceed the number of observations. If a high number of rules is necessary, for example when the data is either very sparse or complicated, a higher number of MCMC-samples is necessary, to substitute for the poor mixing. This tradeoff is limiting, as usually a high number of rules is necessary to have a high enough chance of finding good rules.

Another consideration is to adjust the prior distribution

$$\beta \sim \mathcal{C}(0, \eta) \tag{3.28}$$

in case of logistic regression. Li & Yao [42] recommend using small values ($\eta < 0.1$) for the Cauchy distribution in logistic regression. They observe that for higher values the posterior

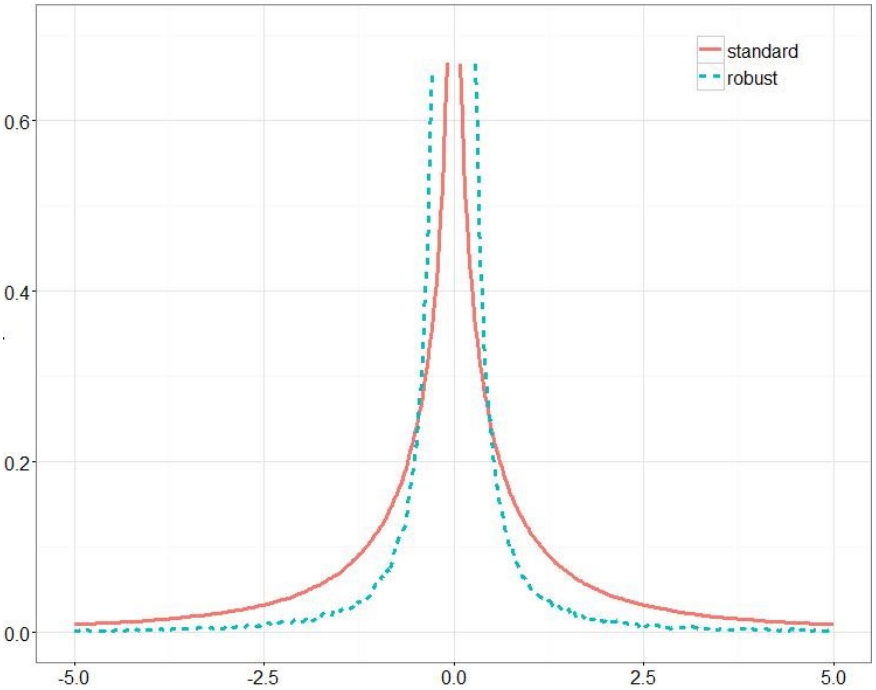


Figure 3.1: Monte Carlo simulated prior density $p(\beta)$ for $\tau \sim C^+(0,1)$(standard) and $\tau \sim C^+(0,0.01)$(robust). The robust prior puts less mass on unreasonable values.

distribution becomes close to the likelihood. This is not desired, as the MLE goes to $\pm\infty$ in the case of separation. They report in their experiments, that small scale values still allow signals to become large [42]. In the Horseshoe hierarchy there is no analytic form available for the marginal prior distribution $p(\beta)$ [15]. Small scales can thus only be obtained indirectly by adjusting the prior distribution for either λ or τ. As it is the indivduall shrinkage prior λ_j which gives the Horseshoe its high adaptivity and we overall expect β values to be smaller in logistic regression it makes sense to instead adjust the overall shrinkage level τ. Gelman et al. [29] argue that in real data sets β coefficients higher than 5 are extremely rare as it means a change in probability from 0 to 0.99 when fulfilled. Following this reasoning in this work

$$\tau \sim \mathcal{C}^+(0, 0.01), \tag{3.29}$$

is used for classification and referred to as robust Horseshoe prior[1]. Figure 3.1 shows the resulting $p(\beta)$ under the robust prior, using Monte Carlo Simulation (because of the unavailability of the analytic expression). It puts much less mass on unreasonable high values. Li & Yao [42] suggest using a fixed value for τ however in this work τ is sampled. This gives the horseshoe its adaptivity and also frees the user from the need to specify a fixed value. It has to be expected, that these compromise settings limit the ability of the Gibbs Sampler to travel through the modes efficiently, compared to the regression setting. This might lead to deterioration in performance, but is necessary to ensure a reasonable mixing.

Actions Taken: To summarize the actions taken to prevent extreme behavior of the MCMC-chain in classification are:

1. Check if the number of predictors exceeds the number of observations and give out a warning if it does. As a standard choice, only a random subset of the decision rules

[1] It is unrelated with the term robust prior in the literature even though there is some similarity.

is kept so that $n > p$. This is a compromise, as the convergence problems seem to overwhelm the information gain by more decision rules.

2. Instead of the normal Horseshoe hierarchy the novel robust Horseshoe prior is used.

Computational aspect In my experiments the algorithms for probit and logistic regression lead to very similar results. From a computational point of view the probit regression is less demanding. Even though the Polya-Gamma sampling is highly optimized with a rising number of n the computational cost increases more than in the case of the Probit sampler. For the empirical experiments I therefore used the probit regression, as it scales better for bigger data sets. Logistic regression is often desired due to its interpretation through the log odds and can be done without a big loss in efficiency.

3.2.4 Hyperpriors

Friedman & Popescu [27] leave the $r_l(x)$ unstandardized, in order to put extra shrinkage on rules with low support. This comes from the Lasso loss function, which penalizes the magnitude of the coefficients. Rules with low standard deviation (low support) have in average higher coefficients and get penalized stronger by the Lasso. Implicitly that incorporates prior knowledge - or more prior wish - to prefer decision rules with high support.
With the Horseshoe regression I chose to standardize the $r_l(x)$, as it is not clear how different scales will effect the complex interplay of the global and local shrinkage parameters λ_l, $l = 1, \ldots, m$ and τ. Even more, the Bayesian framework allows to incorporate the prior information to prefer simple rules *explicitly*. Recall, the prior setting for the local shrinkage

$$\lambda_l \sim \mathcal{C}^+(0, A_l)$$

with $A_l = 1\, l = 1, \ldots, m$ leading to the standard Horseshoe regression. $A_l < 1$ will give a predictor a priori a lower probability of being treated as signal, but the heavy Cauchy tails will allow truly informative predictors to still become big. An analogous to the RuleFit approach can be achieved by giving the decision rules priors proportional to their support

$$A_l \propto s(r_l) = \frac{1}{N} \sum_{i=1}^{N} r_l(x_i). \tag{3.30}$$

This gives rules with low support a priori less chance of being chosen by the model as signal. Another way could be to consider both support and length of the rule. As described before both rules which cover only a small number of observations, as well as decision rules with a lot of conditions are very undesirable for interpretation and may also not generalize well. If a rule consists of only a few observations and is defined by many conditions then it is not very trustworthy, especially in the presence of noise. Such a rule should only be taken into the model as last resort, if no other rule or linear term is able to explain this part of the variation. One way to express this belief is due

$$A_l = s(r_l) \cdot (1 + len(r_l))^{-\eta}, \tag{3.31}$$

where $len(r_l)$ denotes the length of rule l defined as its number of conditions, $s(r_l) \in (0, 1)$ and $\eta \in [0, \text{inf})$ is a parameter controling how strong our prior belief is, that simple rules are sufficient for this data. This is an analogous to the prior probability for a node being non-terminal in BART [17]. One possible way is to set

$$A_l = \sqrt{\frac{2 \cdot \min(s(r_l), 1 - s(r_l))}{\sqrt{len(r_l)}}}. \tag{3.32}$$

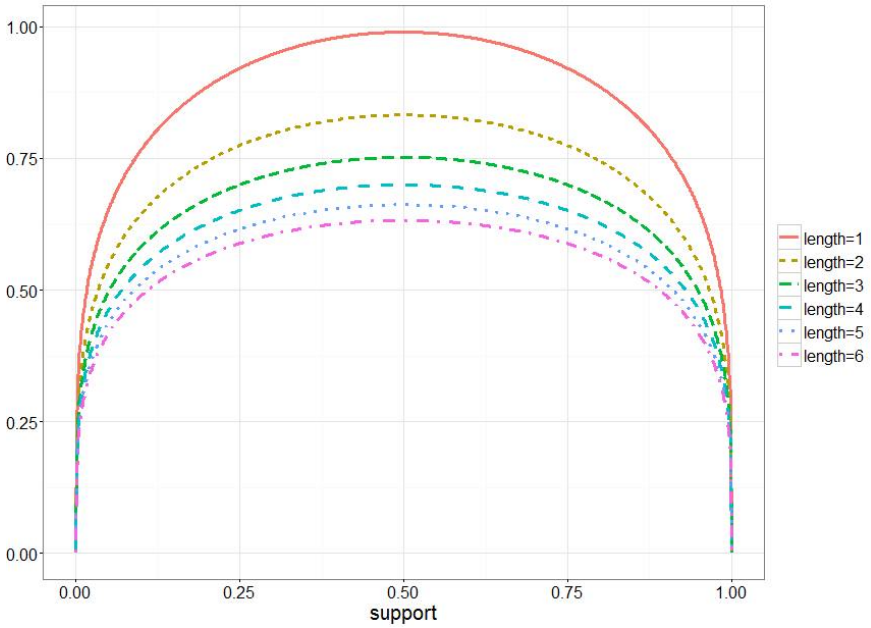


Figure 3.2: The hyperparameter A_l depending on the support of a rule and its length. The function is slowly decaying to give complicated and specific rules still a non-zero prior probability of being used.

The resulting function can be seen in Figure 3.2. A_l is exactly one when the support is 0.5 and the length is one, as the ideal, while rules which are either more complicated or cover less observations get a priori less probability of being signal. The square roots are used, to make the function decay more slowly.
This allows to set the treedepth parameter L (section 3.1.2) in the rule generating process higher, without necessarily overfitting, as through the prior settings simple rules which cover a lot of observations are preferred by the model. However if the data requires it more complex rules might be chosen. This should make the algorithm more adaptive to the data and less dependent on the choice of the tree generating parameters.
An indirect way is to handle A_l as random variable. This leads to the Horseshoe+ hierarachy by [7]. The idea of the horseshoe+ is to include a further mixing variable for the local-shrinkage parameter

$$\lambda_l|\eta_l \sim \mathcal{C}^+(0,\eta_l) \tag{3.33}$$
$$\eta_l \sim \mathcal{C}^+(0,A_l). \tag{3.34}$$

with $A_l = 1, l = 1,\dots,m$ is the Horseshoe+. This extension leads to a stronger push of the noise towards zero and of the signal away from zero. The Horseshoe+ can be integrated in the Horseshoe hierarchy described in section 3.2.1. Using the Inverse-Gamma representation of the half-Cauchy the full-conditional of ν becomes

$$\nu_l|\lambda_l^2,\eta_l^2 \sim \mathcal{IG}\Big(1,\frac{1}{\lambda_l^2}+\frac{1}{\eta_l^2}\Big), \tag{3.35}$$

and the full conditionals of the newly introduced mixing variables are

$$\eta_l^2|\psi_l,\nu_l \sim \mathcal{IG}\Big(1,\frac{1}{\nu_l}+\frac{1}{\psi_l}\Big), \tag{3.36}$$
$$\psi_l|\eta_l^2 \sim \mathcal{IG}\Big(1,\frac{1}{\eta_l^2}+\frac{1}{A_l^2}\Big). \tag{3.37}$$

The derivation can be found in Appendix A. The upper part of the the horseshoe hierarchy remains unchanged [44]. In the context of rule ensembles it is a consideration to replace the non-informative prior $A = 1$ with the prior considerations above. This leads a priori to a stronger gravitation towards zero for complicated and very specific rules. Through incorporating the prior knowledge deeper in the hierarchy very good rules still can escape and become big if the model needs it. Instead of dominating the model, the prior should only guide the search for good predictors. Incorporating the informative prior deeper in the hierarchy gives the model more flexibility to overwrite it if necessary.
Especially important is to adjust the prior for the linear terms as well, as they will be overrepresented otherwise. I chose to set the prior for linear terms to 0.875, which corresponds to the prior of a rule with rule with support of 0.4 and a length of 1.5. Other choices would be possible and should be tested in future work. Rules with support around 0.5 and one condition will have values around 1, while more complicated rules have a priori a lower chance to become big. A is limited at 1, as higher values would force predictors into the model.
This use of the Horseshoe+ to incorporate prior knowledge is rather creative and not studied in the literature yet. Given the specific setting of the rule ensemble in my opinion it makes sense to use all prior knowledge available, in this case about the rule structure.

3.2.5 Implementation in R

As stated above the available implementation of the horseshoe regression in R does not scale well for large p. Therefore I used own code for the Gibbs-Sampling, inspired by the matlab code provided by Makalic & Schimdt [44]. Sampling the Polya-Gamma variables is done with the package BayesLogit by Polson et al. [52] and the rest based on own implementations.

3.3 Posterior Summary

3.3.1 Thresholding

One problem in the context of shrinkage priors is to find a sparse summary of the posterior distribution. The Horseshoe prior will shrink unimportant covariates, but not set them to exactly zero as the Lasso.
The easiest way to address this problem is to define a threshold t and remove all covariates whose $\beta_j < t, i = j, \ldots, p$. This approach creates the need of defining an appropriate threshold, which only removes the covariates which do not contribute to the models performance. Defining such a threshold a priori is a challenging task. Depending on the number of informative and uninformative predictors the magnitude of the β changes. If one adds informative predictors to the model the overall shrinkage τ will increase and in average β will decrease, as the model is now based on more predictors.
Carvalho et al. [15] recommend instead of thresholding the β directly use $\omega = \mathbb{E}(\frac{1}{1+\lambda} > 0.5)$. This can be interpreted as a pseudo-inclusion probability of a covariate. Covariates with $\omega > 0.5$ are signal and $\omega < 0.5$ are noise. Including all and only covariates with inclusion probability higher than 0.5 was shown to minimize squared error prediction loss under orthogonality [6]. However when predictors are correlated this approach might lead to poor model choices. When there is high correlation between predictors the predictors will swap place in the model from time to time. This will decrease the individual κ_j. In a extreme case of many correlated predictors all of the κ_j values might fall under 0.5 and would be removed as noise. In less extreme cases, removing one of the correlated predictors, will leave a hole. Either the model has to be rerun to allow other predictor to fill that spot, or another predictor has to be "unshrunk" accordingly.

3.3.1.1 DSS Posterior Summary

Decoupling Shrinkage and Summary(DSS)[35] addresses the problem of finding sparse posterior summaries in the context of linear models under shrinkage priors such as the Horseshoe. The main idea is to decouple Shrinkage and Summary. The Model fitting procedure remains unchanged and after the posterior predictive distribution

$$f(\hat{Y}) = \int f(\hat{Y}|\beta\sigma^2) f(\beta, \sigma^2|\boldsymbol{Y}) d(\beta, \sigma^2) \tag{3.38}$$

is obtained, sparsity is induced by minimizing the cost function

$$\mathcal{L}(\hat{Y}, \gamma) = \lambda \|\gamma\|_0 + n^{-1} \|\boldsymbol{X}\gamma - \hat{Y}\|_2, \tag{3.39}$$

for a given value of λ. $\|\gamma\|_0 = \sum_j I(\gamma \neq 0)$ denotes the count norm and $\|X\gamma - \hat{Y}\|_2$ is the euclidean norm. The DSS cost function explicitly trades of parsimony for predictive accuracy. The first term measures the model complexity, while the second term measures the degradation in predictive performance using the sparsified solution γ. By that equation (3.39) tries to reconstruct the solution obtained from the predictive posterior distribution with least covariates as possible. DSS decouples the problem of finding a good summary in the two steps, one statistical and one pure optimization. The sparsified solution is obtained by

$$\beta_\lambda = \arg\min_\gamma \lambda \|\gamma\|_0 + n^{-1} \|\boldsymbol{X}\bar{\beta} - \boldsymbol{X}\gamma\|_2 \tag{3.40}$$

The solution is dependent on the choice of λ which expresses how much predictive accuracy one is willing to sacrifice in order to achieve better interpretability. This optimization procedure not only sets coefficients to zero, but also unshrinks coefficients if necessary, in the case of correlated predictors. Setting $\lambda = 0$ will give the posterior mean $\bar{\beta}$ as solution, while large values of λ lead to the null model. Which value is appropriate to fulfill the users need depends on the problem by hand as well as the data set. Hahn & Carvalho [35] recommend to monitor two different measures, to guide the search of a suitable λ value. In analogous to the variance explained in ordinary least squares regression

$${\rho_\lambda}^2 = \frac{n^{-1} \|\boldsymbol{X}\beta\|^2}{n^{-1} \|\boldsymbol{X}\beta\|^2 + \sigma^2 + n^{-1} \|\boldsymbol{X}\beta - \boldsymbol{X}\beta_\lambda\|^2} \tag{3.41}$$

allows to observe the deterioration of the model fit on the training data, when using the sparsified solution β_λ. Note that in (3.41) β is a random variable therefore also ${\rho_\lambda}^2$ becomes a random variable for which credible intervals can be calculated. On the other hand the excess error

$$\psi_\lambda = \sqrt{n^{-1} \|\boldsymbol{X}\beta_\lambda - \boldsymbol{X}\beta\|^2 + \sigma^2} - \sigma \tag{3.42}$$

gives an estimation of the amount of error in future predictions one has to expect due to the sparsification. Again ψ_λ is a random variable for which credible intervals can be obtained. To find a good value of λ it is helpful to plot the model size against ρ_λ and against ψ_λ . One rule of thumb by [35] is to take the sparsest model which still contains $\rho_{\lambda=0}$ and $\psi_{\lambda=0}$ in their 90% credible intervals.

4 Results

This chapter evaluates the performance of the Hs-RuleFit in relation to competing methods. In the first part the predictive performance is examined on a variety of real data sets for regression, classification and genetic expression data. The latter is also a classification task but somewhat special, through the small sample sizes ($p >> n$), the sparsity of the data and spurious relationships.
As stated above, predictive performance is usually not the only goal, but important to give confidence in a model. The second part of the empirical evaluation will examine if Hs-RuleFit is capable of producing easily interpretable models. In previous studies these two steps are often combined and interpretability measured through model complexity, for example in [1]. Automatic model sparsification is not straightforward using the DSS approach described in Section 3.3.2. The right amount of shrinkage is a tradeoff between accuracy and simplicity and should be carefully chosen. It is also dependent on the data, therefore ad-hoc approaches such as limiting the model size to $p = 30$ are not realistic. Two data sets are analyzed in detail to showcase the capability of Hs-RuleFit to produce interpretable models.

4.1 Regression

This section gives a comparison of Hs-RuleFit with competing methods on a variety of real regression data sets. The 16 data sets used for comparison are a subset of the data sets used in [38] and [17]. The choice of data sets was guided by public availability not by its characteristica. Table 4.2. gives a summary of the used data sets. N denotes the number of Observations, Q the number of quantitative covariates and C the number of categorical covariates. This selection represents a variety of size, dimensionality and problem difficulty. For this reason the algorithms are compared using the relative root mean squared error (RRMSE)

$$RRMSE((x,y),F_k) = \frac{RMSE_k((x,y),F_k)}{\underset{j\in K}{min} RMSE((x,y),F_j)}, \tag{4.1}$$

Table 4.1: The Settings for the used methods.

Method	Parametersetting
Hs-Rulefit	Number of Trees: 1000 L:4 Ensemble: GBM
Hs-Rulefit+	Number of Trees: 1000 L:5 Ensemble: GBM+RF
RuleFit	Number of Rules: 2000 L:4
Random Forest	Number of Trees: 500 Variables tried: $\sqrt{p}$
BART	$(\gamma, q) = (3, 0.9)$ μ prior:2 Number of Trees: 200

Table 4.2: The 16 Regression Data sets.

Name	N	Q	C
Abalone	4177	7	1
Ais	202	11	1
Boston	506	13	0
Budget	1729	10	0
Cps	534	7	3
Cpu	209	6	1
Diamond	308	1	3
Hatco	100	6	4
Heart	200	13	3
Insurence	2182	4	2
Medicare	4406	21	0
Mpg	392	6	1
Ozone	330	8	0
Servo	167	2	2
Strike	625	4	1
Tecator	215	10	0

of model F_k and K is the set of compared algorithm. The root mean squared error (RMSE) is defined as

$$RMSE((x,y),F_k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - F_k(\boldsymbol{x_i}))^2}. \tag{4.2}$$

The RRMSE accounts for different scales of y, as well as different difficulty as it measures the relative performance. On each data set 10-fold cross-validation is performed. In each split 9-folds are used for model building and one fold left out for prediction. For comparability with other studies also the RMSE is reported.
The compared algorithms are Random Forest, BART, RuleFit, Hs-RuleFit and Hs-RuleFit+. Hs-RuleFit uses *only* Boosting generated rules and the standard Horseshoe regression. This allows a direct comparison with RuleFit to assess the performance of the Horseshoe prior and the Lasso. Hs-RuleFit+ uses both Boosting and Random Forest generated rules together with the rule-structure prior specifications from Section 3.2.4.. BART is chosen for comparison, as it shares a similar idea to Hs-RuleFit and the Random Forest as a strong default choice algorithm. BART and Random Forest were built using the R packages bartMachine and randomForest respectively. No dummy coding of the categorical variables is necessary, as the implementations can deal with categorical variables internally. RuleFit was built using the package provided by [27]. For the Hs-RuleFit I used my own code in R.
No cross-validation for parameter tuning was performed for any of the algorithms. Instead the default settings shown in Table 4.1 were used. This should allow a fair comparison, as all algorithms could be tuned for a better performance.
All numerical covariates are scaled and y is centered. This has no effect on the tree based methods Random Forest and BART which are scale invariant, but is important for the rule ensemble methods, which use linear terms as well.

Results

Figure 4.1 shows the boxplot for the RRMSE of the $16 \cdot 10 = 160$ splits over all data sets. Hs-RuleFit+ and Hs-RuleFit outperform its competitors overall by a wide margin. The Hs-RuleFit performs significantly better than RuleFit. Therefore the empirical results confirm the theoretical expectation that the Horseshoe prior is better suited for the task of Rule Ensembles than the Lasso. There is minor differences in the implementation, e.g. the RuleFit uses Huber-Loss on the Regression, but they cannot account for the substantial difference in

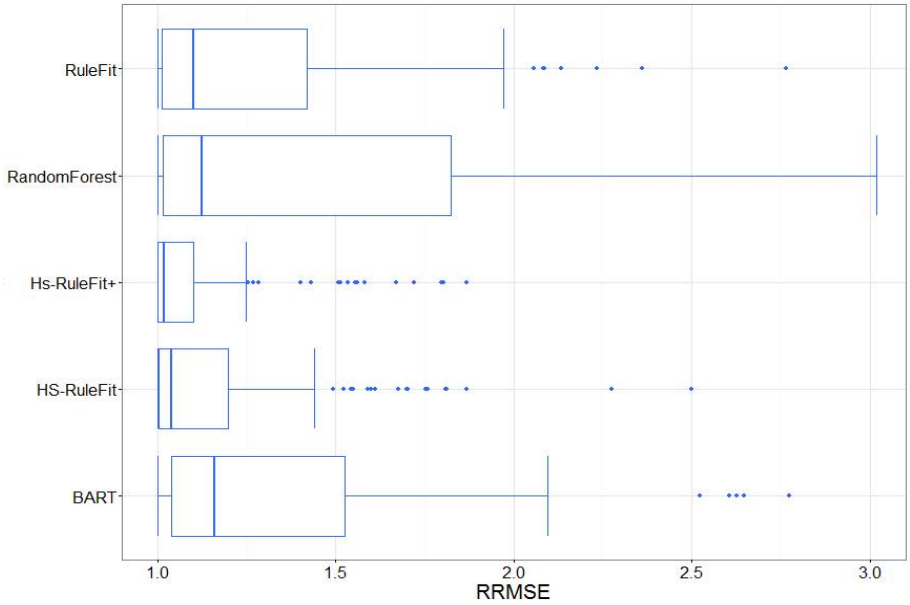


Figure 4.1: Boxplots showing the RRMSE across the 160 train/test partitionings over the 16 data sets. There is still a considerably amount of values higher than 3 for Random Forest (12.5%) and BART (6.1%)

accuracy.
The clear winner in this experiment is the Hs-Rulefit+. The increased adaptivity through the mixture of tree ensembles in the rule generation, as well as explicitly using prior information about the rules in the model fitting process, lead to superior predictive performance. Using the rule-structure prior allows to grow deeper trees with $L = 5$ without the danger of overfitting. Looking at Figure 4.1 Hs-Rulefit+ has much tighter intervals and only a few splits in which it is significantly worse than the best algorithm. This shows that it is able to adapt to very different data sets of different complexity.
BART outperforms Random Forest overall, which confirms the results in [17]. Interestingly the RRMSE median for Random Forest is lower. This means, that there is data sets where Random Forest performs well compared to the other methods. Table 4.3 shows the RMSE results on the different datasets, together with a ranking on the dataset. In fact Random Forest is the best method in two data sets. BART has overall the worst ranking in this experiment, but the average RRMSE is significantly better than the average RRMSE of Random Forest. Reason for this is, that in datasets in which BART performs well, the rule ensemble methods perform even better. That speaks for a similarity between BART and the rule ensemble methods, while the working of Random Forest is fundamentally different. One reason might be, that all methods except Random Forest are, to a certain degree, based on Boosting. Interestingly the scores of BART and Hs-RuleFit+ are the most different ones. This is suprising, as both apply bayesian methodology and the prior specification in Hs-RuleFit+ is inspired by BART. An explanation could be the in [17] stated characteristic of BART to model spurious relationships. In some data sets, especially when there is no clear pattern, such as very noisy data like the Heart data set this leads to good performance as it puts less confidence on single explanations. Hs-RuleFit+ works orthogonal to this. Spurious relationships get removed aggressively, through the Horseshoe prior. When there is a clear pattern Hs-RuleFit+ tends to perform better, as it is not distorted by spurious relationships. Looking at individual data sets confirms this hypotheses. The Tecator and the Budget Data set contain highly correlated predictors. Notable these are the data sets where the difference between BART and Hs-RuleFit+

Table 4.3: 10-fold Cross validation results over the 16 regression data sets. Each entry shows the RMSE and in parentheses the rank on this data set. The best result is marked in bold.

	Random Forest	BART	RuleFit	Hs-Rulefit	Hs-Rulefit+
Abalone	2.143 (3)	2.147 (4)	2.156 (5)	2.137(2)	**2.129** (1)
AIS	0.097 (5)	0.080 (3)	0.085 (4)	0.055 (2)	**0.055** (1)
Boston	3.157 (2)	3.324 (4)	3.300 (3)	3.470 (5)	**3.054** (1)
Budget	0.075 (4)	0.179 (5)	0.037 (3)	**0.025** (1)	0.026 (2)
CPS	4.362 (3)	4.331 (2)	**4.323** (1)	4.383 (4)	4.593 (5)
Cpu	0.359 (3)	0.400 (5)	0.372 (4)	0.353 (2)	**0.352** (1)
Diamond	1210.8 (5)	319.55 (4)	255.53 (2)	259.52 (3)	**193.39** (1)
Hacto	0.343 (5)	0.259 (3)	0.257 (2)	0.300 (4)	**0.250** (1)
Heart	**27.938** (1)	28.370 (3)	28.147 (2)	31.852 (5)	30.880 (4)
Insurence	0.702 (5)	0.654 (3)	0.657 (4)	**0.633** (1)	0.641 (2)
Medicare	11.614 (4)	12.045 (5)	**11.543** (1)	11.586 (2)	11.611 (3)
MPG	2.727 (2)	2.968 (5)	2.784 (4)	2.739 (3)	**2.701** (1)
Ozone	**3.975** (1)	4.044 (2)	4.098 (3)	4.111 (4)	4.328 (5)
Servo	0.453 (5)	0.311 (4)	0.297 (3)	0.261 (2)	**0.260** (1)
Strikes	0.702 (5)	0.654 (3)	0.657 (4)	**0.633** (1)	0.641 (2)
Tecator	0.542 (3)	0.604 (5)	0.563 (4)	0.438 (2)	**0.427** (1)
Average Rank	3.50	3.75	3.06	2.69	**2.00**

is the highest. The BART model is distorted by modeling spurious relationships, while Hs-RuleFit is able to deal with them efficiently.
RuleFit performs suprisingly well on these datasets. This is in contrast to the findings in [1], where Random Forest performs better than RuleFit. It should be noted that in [1] $L = 6$ and 5000 rules are used. The higher treedepth is likely to lead to overfitting and the high number of rules leads to suboptimal behavior of the Lasso as discussed previously. Using the recommended standard settings RuleFit performs well, which is an indicator that the idea of simplifying tree ensembles into rule ensembles can indeed increase predictive performance. However the difference in performance also shows the sensitivity of RuleFit to its settings.
On 12 of the 16 data sets the Hs-Rulefit methods perform the best and have tight intervals, indicating high stability of the produced models. The Hs-RuleFit+ improves both accuracy and stability even further compared to Hs-RuleFit. This can be partly contributed to the usage of Random Forest generated Rules. In the Boston and the MPG Data set where the difference between Hs-RuleFit and Hs-RuleFit+ is large Random Forest performs well, being the second best method. Random Forest seems better on this data sets in finding informative subspaces. On the other hand in the Diamond and Hatco Data the difference is quite large as well, while Random Forest is by far the worst method on these Data sets. This indicates that the incorporation of the rule structure as prior information gives the model improved accuracy and stability as well. A last interesting observation is that the difference between Hs-RuleFit and Hs-RuleFit+ is the biggest on the small data sets. The rule structure prior specification seems to help on these small data sets to prevent overfitting, which is a smaller problem in the bigger data sets.

Table 4.4: The eight classification data sets used for evaluation.

Name	N	Q	C	Name	N	Q	C
Australia Credit	690	6	8	Haberman	306	3	0
Diabetes	768	8	0	Heart	270	9	4
Ecoli	336	4	2	Hepatitis	155	6	12
German Credit	1000	7	13	Ionosphere	351	34	0

Table 4.5: 10-fold Cross validation results over the 8 classification data sets. Each entry shows the average misclassificationrate and in parentheses the rank on this dataset. The best result is marked in bold.

	Random Forest	BART	RuleFit	Hs-Rulefit	Hs-Rulefit+
Australia Credit	**0.128** (1)	0.145 (2)	0.134 (3.5)	0.145 (5)	0.134 (3.5)
Diabetes	**0.237** (1)	0.240 (3)	0.239 (2)	0.261 (5)	0.254 (4)
Ecoli	**0.032** (1)	0.033 (2.5)	0.042 (5)	0.035 (4)	0.033 (2.5)
German Credit	**0.238** (1)	0.248 (4)	0.249 (5)	0.247 (3)	0.24 (2)
Haberman	0.272 (2)	**0.268** (1)	0.282 (3)	0.288 (4)	0.30 (5)
Heart	0.314 (4.5)	0.295 (2)	0.300 (3)	0.314 (4.5)	**0.291** (1)
Hepatitis	0.291 (1.5)	0.291 (1.5)	0.400 (5)	0.341 (4)	0.332 (3)
Ionosphere	0.068 (2)	0.08 (5)	**0.065** (1)	0.072 (3.5)	0.072 (3.5)
Average Rank	1.75	2.65	3.38	4.125	3.06

4.2 Classification

In this section the performance on eight classification data sets is compared. Due to the data augmentation the computations are more expensive. Therefore in this study only eight data sets for classification were compared. However the eight data sets are of varying size and dimensionality. Table 4.4. shows the characteristics of the data sets which are all taken from the UCI machine learning repository [43].
The compared methods are the same as in the Regression setting (Section 4.1), all run with standard settings. Only exception is the Hs-RuleFit+ which now uses the robust horseshoe prior described in Section 3.2.3. For comparison the misclassification rate defined as the number of misclassified observations divided by the total number of observations is used for comparison. Again 10-fold cross validation is performed, as described in the previous chapter.

Results

The result on the classification data sets are very different to the regression results discussed previously. The tree ensemble methods perform significantly better overall, with Random Forest being the clear winner of this experiment. Random Forest has the best rank in four of the eigth data sets and performs reasonable on all other data sets as well. Interestingly BART performs worse than Random Forest. This is in contrast to [17] who compared BART and Random Forest in Regression as well as to the findings in the regression experiments. The BART implementation uses the Albert & Chib algorithm which might not be optimal and could explain the different performance.
In the group of Rule Ensemble methods the Hs-RuleFit+ performs better than RuleFit, and Hs-RuleFit is the worst method on these data sets. A likely explanation for this are the in Section 3.2.2 discussed problems of the Horseshoe prior in the presence of separation. Checking of traceplots clearly showed that the MCMC algorithm did not converge in many of the data sets using the Hs-RuleFit. Instead slow mixing behavior and posterior means for β up

to magnitudes of 200 could be observed.
The more robust settings in the Hs-RuleFit+ helped to improve convergence. Visual inspection of traceplots showed better mixing behavior in most of the data sets. However the mixing was still visibly worse than in regression. The improvement in mixing is reflected in an increase in performance, especially on some of the small data sets which are easily separated, Heart and Hepatitis. On bigger data sets the difference is smaller, but Hs-RuleFit+ performs better here as well.
Looking at the exact values in Table 4.4. the differences between the algorithms are overall noteworthy but smaller compared to regression. In most data sets the difference between the best algorithm and the worst is around 2%. Hs-RuleFit+ is at most 4% worse than the best algorithm on the Hepatitis data set. The difference between RuleFit and the best algorithm on this data set is 11%. Hs-RuleFit+ seems to be a bit more stable than RuleFit, however more data sets should be evaluated to verify this hypothesis.
The results clearly indicate that the rule ensembles, despite their superior performance in regression, do not work as well on classification. To my knowledge no systematic studies have been conducted to explain the different performance of rule ensembles in regression and classification. However the here presented results are in concordance with the literature. In the experiments in [21] RuleFit shows the worst performance in classification compared to other rule learners and ensemble methods.

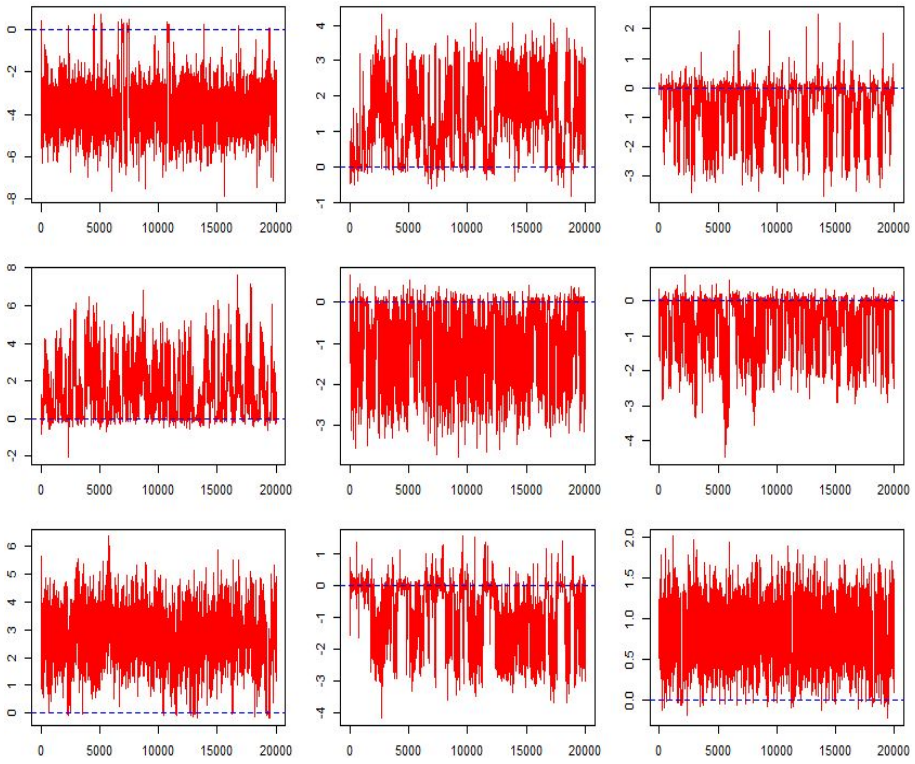


Figure 4.2: The Traceplot for the 9 largest coefficients of Hs-Rulefit on the Boston Housing data.

4.3 Applications

4.3.1 Social Science - Boston Housing Data

The Boston housing data available on the UCI machine learning repository [43] is widely used in machine learning literature. It is also used in the RuleFit paper [27]. The Boston housing data consists of $N = 506$ observations which are city areas in Boston. $p = 13$ variables are recorded. These variables include ecological measures of nitrogen oxides (NOX), particulate concentrations (PART) and proximity to the Charles River (CHAS), the socio-economic variables proportion of black population (B), property tax rate (TAX), proportion of lower status population (LSTAT), crime rate (CRIM), pupil teacher ratio (PTRATIO), proportion of old buildings (AGE), the average number of rooms (RM), area proportion zoned with large lots (ZN), the weighted distance to the employment centers (DIS) and an index of accessibility to key infrastructure (RAD). The dependent variable is the median housing value of the area. For this data set the Hs-RuleFit+ with default settings (L=5, ntree=1000) is used. 90% of the data is used for training and 10 % left out to check if the found sparse solution is reliable.

To monitor the convergence it makes sense to look at the coefficients of the most influential predictors. Figure 4.2 shows the traceplots of the 9 most important coefficients. The importance of the coefficients is measured as the overall impact on predictions which is given as the posterior mean weighted with its standard deviation. Generally in Bayesian regression the plot would speak for a poor convergence, as some of the chains are not randomly moving around one non-zero value, but also have considerable mass at zero. In the context of the Horseshoe prior it indicates multimodality as discussed in Section 2.5. The traceplot indicates, that the model visits different modes, which is desired. Therefore the traceplot speaks for a good convergence.

Using the DSS-Sparsification leads to a model with 51 non-zero predictors with almost iden-

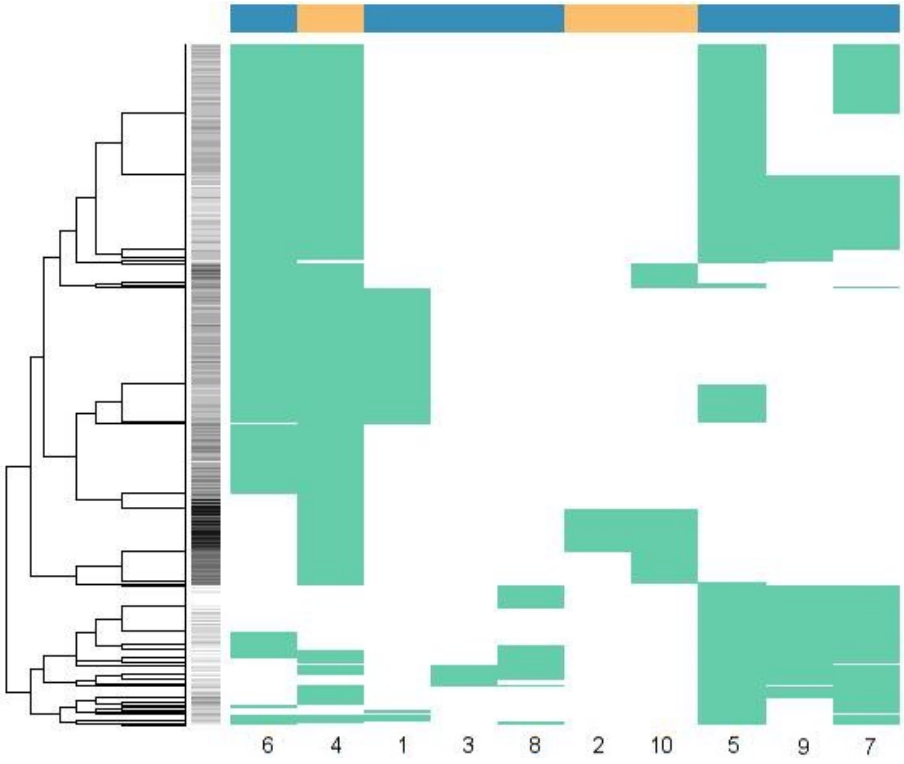


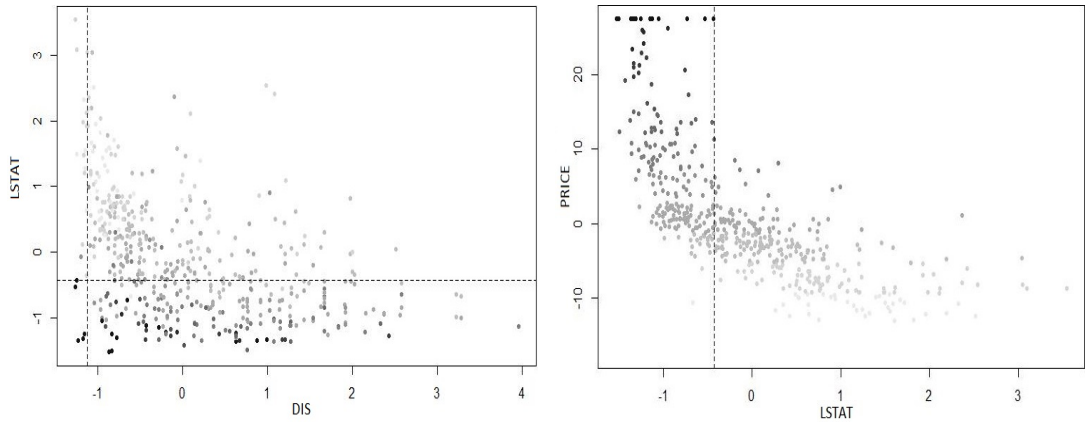
Figure 4.3: RuleHeat for the Boston Housing Data.

tical fit on the training data. Interestingly the test error reduces slightly from 2.89 to 2.86. As a reference, the RuleFit model for this data split consists of 243 non-zero terms and has a test error of 3.44. For this dataset both accuracy and interpretability are clearly better using Hs-RuleFit+.

One tool I found useful to get an insight about the decision rules is to draw a heatmap of the most important *Decision Rules*. This use is novel and is called RuleHeat in the following. The rows represent the observations, with the greyscale indicating their y-value. In this case darker stands for more expensive areas, while the cheaper areas are colored white. The columns represent the 10 most important decision rules, with the column color indicating the direction of the coefficient (blue for negative, sand for positive). In the actual heatmap each area represents $r_j(x_i)$ and is green for $r_j(x_i) = 1$ and white otherwise. Therefore it shows if a rule "fires" for an observation. Also it is possible to find similar observations, through the dendogram on the rows [1]. The RuleHeat allows to see which decision rules describe which groups of observations. Looking at the group of very expensive houses (dark rows), we can see that they are captured through Rule 10: $RM > 0.8$ & $RAD \leqslant 0.7$. Note that the data is scaled. Expensive houses share therefore the characteristic that they have an above average number of rooms and are not very well accessible from highways. Without knowing any details about Boston geographics it makes sense, as expensive houses are often a bit separated from the main city and not that well accessible. Rule 2: $RM > 1.7$ & $PTRATIO \leqslant 0.7$, separates the expensive areas from the very expensive ones. This means that areas with very big houses and a low pupil to teacher ratio are especially expensive. On the other hand cheap areas are characterized by Rule 8:$CRIM > 0.7$ & $NOX > 0.8$, which means high crimerate and strong air pollution. The overall most important rule on this dataset is Rule 1: $RM \leqslant 0.6$ & $DIS > 0.3$, not too big houses which are a bit outside the city. This describes perfectly the suburbs of Boston which are mostly average in their housing prices.

[1] The clustering is only based on the 10 most important rules as well.

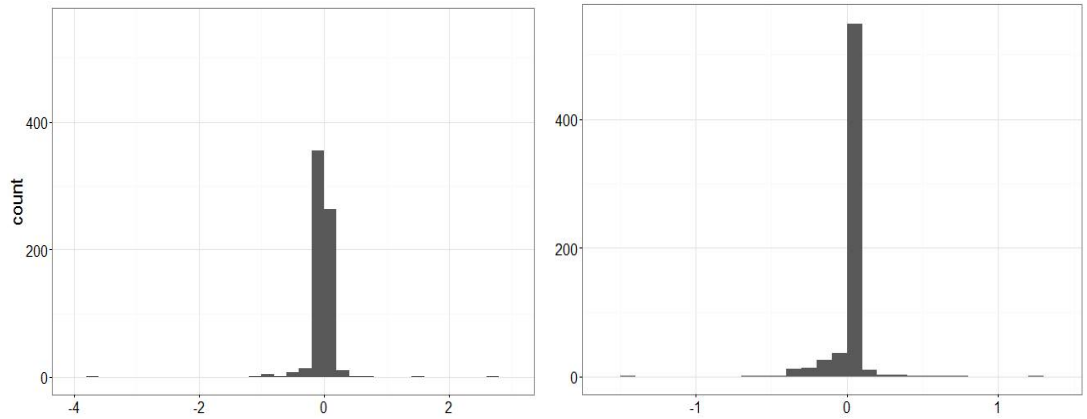
In my opinion the Hs-RuleFit finds very natural clusters with respect to the housing prices and accurate rules describing these clusters. It is also possible to inspect the rules further by looking at the subspaces they define.



(a) Rule 3:$DIS \leqslant -1.12$ & $LSTAT > -0.43$. Rule indicates Top Left quadrant.

(b) Rule 5: $LSTAT > -0.42$. Rule indicates right area.

Rule 3 describes very cheap houses in the city center. Also it shows that there is a good separation for expensive houses in the city center, which seem to be segregated from lower status population. This information could be used in further models. Rule 5 shows the relationship between LSTAT and the housing price. The relationship is clearly non-linear but could possibly be well approximated with polynomials. One could use this information and rerun the model with an appropriate term. In this low dimensional dataset it would have been possible to spot this relationship by standard exploratory data analysis. Note however that the Hs-RuleFit+ found this important variable automatically. This means that it can be very well used in high dimensional data as a data exploration tool, which gives an educated guess about the underlying mechanism and can guide further modeling.
The distribution of the coefficients before and after sparsification is interesting as well.



(a) Histogram for the posterior means of β.

(b) Histogram for β after DSS-Sparsification.

Visibly the sparsification sets many coefficients that are close to zero to exactly zero. But also the range changes. Before using the DSS-sparsification the largest coefficients are -4 and 2.5, after DSS -1.5 and 1.3. A possible explanation is that in the full model the big coefficients get balanced out by many small coefficients, while after simplifying it consists of *individual* informative medium sized coefficients. This corresponds with the natural clusters discussed above.

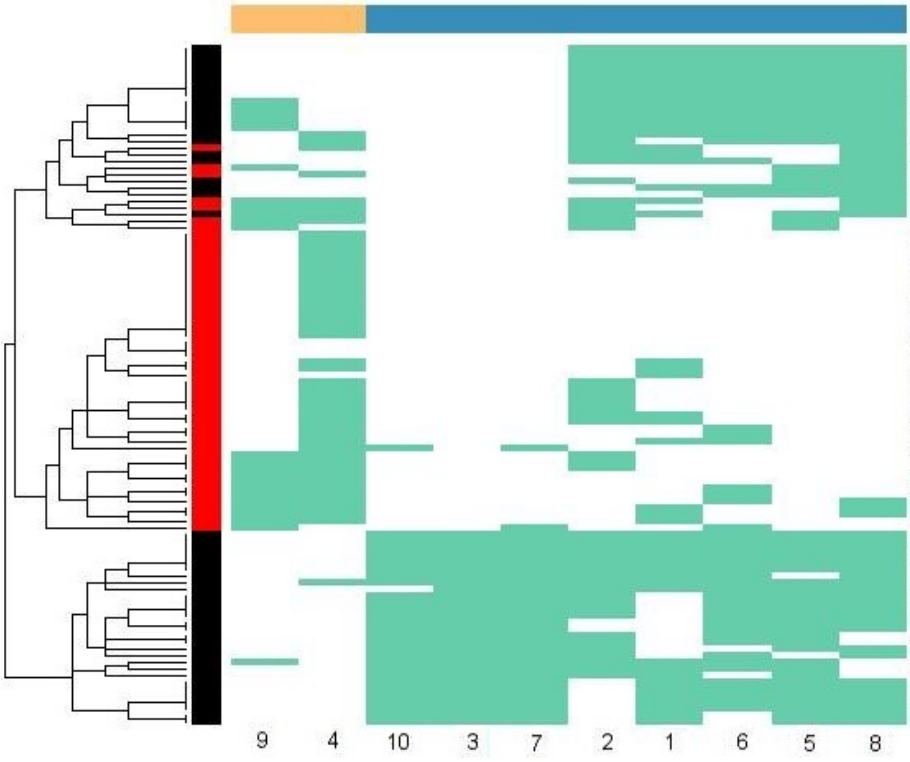


Figure 4.6: RuleHeat for the Prostate Cancer Data set.

4.3.2 Medical - Prostate Cancer Data

One application of statistical learning that successively becomes more important is the analysis of gene expression data. Gene expression is usually characterized by an extreme $p >> n$ situation, due to the huge number of candidate genes and the high cost of obtaining samples. The challenge is to find genes which are overexpressed for patients with a certain kind of cancer, from this large number of candidate genes. Also due to the measurement the data is potentially noisy, systematically biased and gene expressions are highly correlated. By that, even though a classification task its a very different problem from the above described data sets.

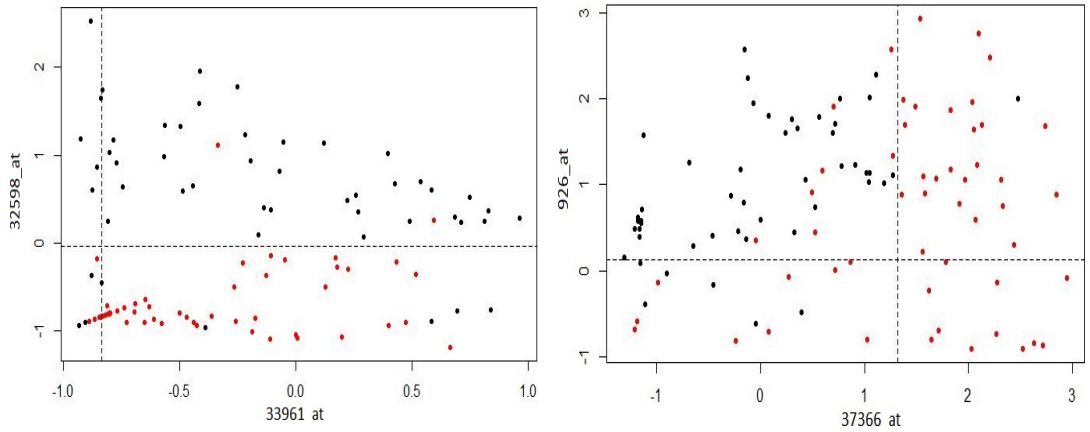
The often used Prostate Cancer data set by [59] consists of 102 gene expression tissues, of which 52 are normal and 50 are cancerous. 12600 genetic expressions are measured for each sample. After using the minimal preprocessing by [59] $p = 5966$ genetic expressions remain for the analysis. The Hs-RuleFit was used with $L = 3$ and Random Forest generated rules. Reason for this is: with the extreme ratio of n to p good separation in more than 2 variables is very likely to occur just by chance and is hardly verifiable with the small number of observations, therefore simple rules are to prefer. Random Forest was chosen for the rule generation because Boosting is computationally extremely expensive in this high dimensionality. Linear terms were also excluded in the initial model, but will be considered in the model build upon the graphical inspection of the Hs-Rulefit model called Hs-Selected.

Table 4.6: 10-fold Cross validation results on the Prostate Cancer data set.

	Random Forest	BART	RuleFit	Hs-Rulefit	Hs-Selected
Average Error in %	10.81	12.83	13.87	6.81	3.82

The Hs-RuleFit performs very well on this data set. The 10-fold Cross-validated Error rate

is significantly lower than for Random Forest, BART and Rulefit run with standard settings. Typically of interest are the most important genes found. For this the Hs-RuleFit was built on the whole training data. Again as a first step one can take a look at the Ruleheat to get an insight about the structure of the found rules. Figure 4.6 shows the Ruleheat for the ten most important rules with red rows indicating the cancer tissues. Interestingly the normal tissues are separated into two groups, where one groups is easily separated with Rule 3: 38406_f_at > 3 while the other group is not that clear and requires many rules to be separated. This might be an indicator that there is two groups of normal patients, where one is characterized as risk group. Gene 38406_f_at was found in previous studies to be associated with cancer, however the risk group interpretation is novel and could be used for future investigation. Rule 5:37639_at $\leqslant 0.92$ seems to overall discriminate best between cancer and normal tissues. Another rule giving good separation is rule4: 33961_at > -0.83 & 32598_at $\leqslant 0.035$. This rule consists of two conditions. As described previously in the extreme $p >> n$ setting one should be cautious about interactions as they can occur just by chance in the training data. In fact looking at Rule 3, I would argue that



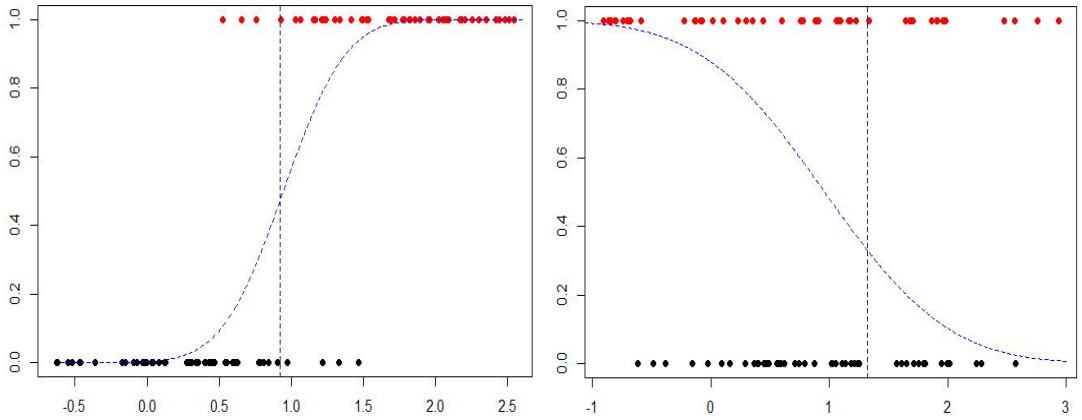
(a) Rule 3: 33961_at > -0.83 & 32598_at $\leqslant -0.035$. Rule indicates Bottom right quadrant.

(b) Rule 5: 37366_at $\leqslant 1.32$ & 926_at > 0.13. Rule indicates top left quadrant.

the split 32598_at $\leqslant 0.035$ seems to be important, while the split 33961_at > -0.83 seems not really justified by the data and could very well just capture random fluctuation. On the other hand rule6: 37366_at $\leqslant 1.32$ & 926_at > 0.13 gives a good separation and there seems to be a tendency in the data, that cancer patients have low values in 37366_at and higher values in 926_at. This interaction could be a candidate to look into in further research. The Hs-RuleFit also found more complicated rules with 3 or more conditions, but as stated above the data is too small to justify this complicated rules. The most important rule found in this dataset is Rule 1: 33677_at $\leqslant 3$ & 33438_at $\leqslant 1.5$ & 38024_at > -1.2. Looking at the RuleHeat this rule is used by the model to discriminate in between the previously described risk group. It is however questionable if this rule reflects a true mechanism and should be investigated with some caution. One option is to use only the rules which seem to be reasonable from the visual inspection in term of discrimination as well as complexity. Rerunning the model with Rule 3, Rule 4 (using only the first condition), Rule 5 and Rule 6 yields a CV-error of only 4.8%. This indicates that these rules are indeed of high quality and are sufficient to reflect the underlying mechanism.

Another consideration is to include the important Genes as linear terms. Reason for this is, that the hard boundaries defined through decision rules are based only on a small sample. This means that we cannot be too certain about the exact position of the boundary. Looking at Rule 2 and Rule 5 shows that the rules are indeed potentially unstable as small changes in the training data could change the boundary, due to the small number of observations. Including the variable as a linear term transforms the hard boundary into a soft boundary.

Accordingly the model was rerun with the previously found genes as linear terms 37366_*at*, 926_*at* 32598_*at*, 38406_*f*_*at*, 37639_*at* *without* the decision rules and the results reported as Hs-selected. The following graph shows the fitted marginal probabilities compared to the decision rules. It is interesting to note, that for Rule 3 both decision rule and cumulative dis-



(a) Rule 5: 37639_*at* $\leqslant$ 0.92. Blue line shows marginal effect for the linear term ($\hat{\beta} = 2.962$).

(b) Rule 6 (first condition): 37366_*at* $\leqslant$ 1.32 ($\hat{\beta} = -1.22$).

tribution function discriminate at $x \approx 0.9$ while they do not match for Rule 6. In my opinion the soft boundaries reflect the uncertainty about this data set better. Indeed, rerunning the model with only the selected important Genes included as linear terms reduces the CV-error rate to 3.82%. This result is on par with the best results on this data sets found in the literature. However it was found without any sophisticated data preprocessing and was computationally inexpensive, as the regularization process was only run on the set of rules, never on the full data. The inital model is built in about 2 minutes compared to 10 hours in [42] who apply Bayesian regularization on the full data set and achieve a slightly better error rate of about 3%. On this data set Hs-RuleFit showed to be excellent as a data exploratory tool, that helps to guide further modeling. Both the Hs-RuleFit model and the Hs-selected model achieve a 3% error rate on a separate test-dataset, which indicates that indeed the right genes were found.

4.4 Simulation Study -Rule Ensembles in Classification

Not much is known about the functioning of tree based Rule Ensembles in classification. Two questions are of major concern: (1)How does the Hs-RuleFit work in classification? (2) How do the rule ensembles create a margin between the groups? Question (2) has the following motivation: Both Random Forest and Boosting use resampling schemes to increase the margin between $y = 1$ and $y = 0$ observations. Boosting uses a reweighting (arcing), which gives previously misclassified cases a stronger weight in the next iteration. By that the next rule/tree is tailored to discriminate between the hard to classify cases. Random Forest uses non-parametric bootstrapping (bagging) to obtain a variety of trees. In some of the bootstrap samples the hard to classify cases will dominate, hence the grown tree will be tailored for them. Both resampling schemes increase the margin between $y = 1$ and $y = 0$ by stacking Trees which are specialized on the hard cases. In contrast the Regularization methods aim to find sparse solutions, which is directly orthogonal to the stacking approach. The expectation is that, if possible only a few rules are chosen, that classify the whole set of observations and no special attention is given to the margin.

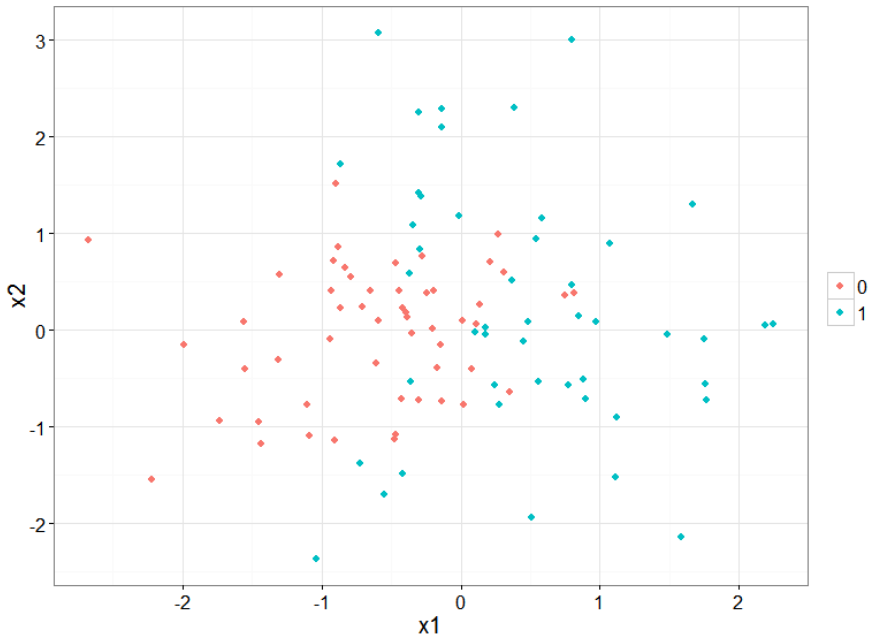


Figure 4.9: The generated data. It is not separated, but complete separated subspaces can be found by Decision Trees.

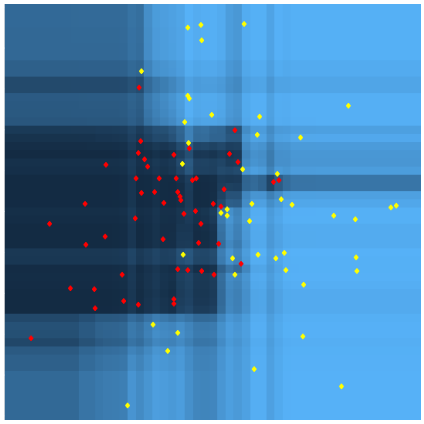
The data is generated with

$$\begin{aligned} x_1, x_2 &\sim \mathcal{N}(0,1), \\ z &= 2 \cdot x_1 + (x_2)^2 + \epsilon, \\ \epsilon &\sim \mathcal{N}(0, 0.5), \\ y &= I(z > 0.5), \end{aligned}$$

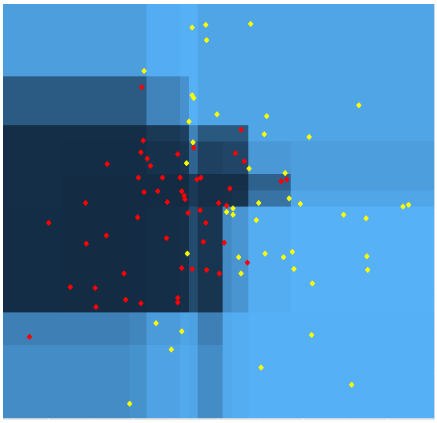
and n=100. This experiments represents the situation with a true underlying latent variable z from which we only observe the dichotomous outcome y. The quadratic term is used to make the function more difficult and non-linear. The following methods are compared:

1. Random Forest
2. Hs-RuleFit with $\tau \sim \mathcal{C}^+(0,1)$
3. Hs-RuleFit with $\tau \sim \mathcal{C}^+(0,0.01)$
4. RuleFit

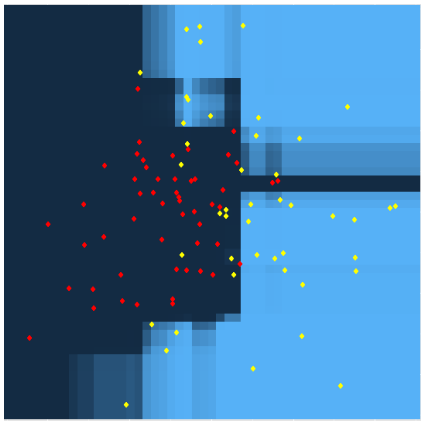
Figure 4.10 shows the fitted probabilities over the covariate space for the different methods. Comparing Random Forest and RuleFit, it becomes clear, that RuleFit uses only few rules to discriminate between the classes. The transition at the boarder is very abrupt, changing from $p \approx 0$ to $p \approx 0.8$. The Rules found by RuleFit seem to discriminate very well on the *training data*. The problem is, that through the hard boundary a new observation changes its probability to be $y = 1$ from 0 to 0.8 by changing marginally in $\mathbf{x}$. Such an apprupt change seems to be not justified by the data, as it greatly overestimates the certainty of the model about cases close to the boundary.
On the contrary under Random Forest the boundaries are defined by many rules. This comes from the previously described special treatment of the boundary through resampling. The transition is very smooth here, and especially the boundary seems to reflect the uncertainty



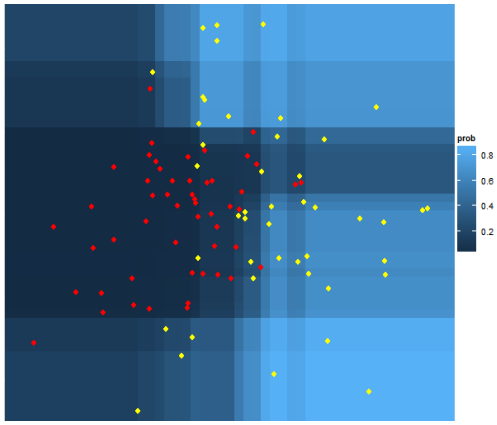
(a) Predicted Probabilities with Random Forest.



(b) Predicted Probabilities with RuleFit.



(c) Predictive posterior mean probabilities under Hs with $\tau \sim \mathcal{C}^+(0,1)$.



(d) Predictive posterior mean probabilities under Hs with $\tau \sim \mathcal{C}^+(0,0.01)$.

Figure 4.10: Comparison of the predicted probabilities of the different methods, darker blue represents probabilities close to zero, while lighter blue indicates higher probabilities. Red points are $y = 0$ yellow are $y = 1$.

well, as the probability gradually decreases.
The Hs-RuleFit with $\tau \sim \mathcal{C}^+(0,1)$ shows problematic behavior. Especially the horizontal stripe with $p = 0$ (dark stripe) is interesting. The corresponding rule separate perfectly in the training data and the posterior mean becomes extremely high. This stripe however does not capture any true underlying mechanism. New cases falling into this rules which are on the right side of the plot will be misclassified. Overall the Hs-RuleFit predicts with an extreme certainty, which is not justified by the data.
Adjusting $\tau \sim \mathcal{C}^+(0,0.01)$ does not allow the coefficients to become unreasonable high and overall captures the true pattern better. It seems that overall the uncertainty increases after adjusting the τ, resulting in many areas having probabilities between 0.4 and 0.6. This might be a direct result from the change of the prior distribution and could effect the predictive performance negatively.
In this experiment the rule ensembles do not capture the uncertainty in the boundary regions well, but seem to put too much certainty on single rules, reflected as extremely high β coefficients. The Hs-RuleFit clearly shows problems caused by separation.

5 Discussion

This chapter discusses the theoretical and empirical results from the previous chapters together with problems encountered in this work. The problems are isolated as much as possible and possible solutions are discussed.

5.1 Results and Implications

Both RuleFit and Hs-RuleFit showed great performance for Regression. Especially the Hs-RuleFit+, which used the Bayesian methodology to incorporate prior information about the rule structure outperformed its competitors heavily. The empirical results together with the theoretical arguments provide evidence that the horseshoe prior is a better choice of regularization than the Lasso for Rule Ensembles in Regression. On the other hand the classification results are not that clear. The Hs-RuleFit+ still performs better than the RuleFit, but both get outperformed by the Tree Ensemble methods. The simulation study implies that the Rule Ensembles put too much confidence on single rules. This partly is due to the separation problem, which effects only the rule ensembles which are based on linear models. RuleFit is less effected but still performance is worse for classification. In my opinion an even bigger problem is the way the "close" cases are handled. Below figure shows the problem in a toy example with five observations.

(a) Boundary when using all data.

(b) Boundaries with Resampling.

When using a resampling procedure the one observation defining the boarder is not used in all trees, leading to different boundaries. By averaging over the different trees and boundaries, as in Random Forest, the uncertainty in this region is captured naturally. On the other hand when fitting rule ensembles which are based on linear models all observations are taken into account. If there is one rule which separates all cases well, there is no need to take more rules into the model. This is especially true when using regularization as we search for a sparse solution. As a consequence close cases have the same estimated probability as the more "sure" cases. The uncertainty in this region is therefore overestimated by the model.

I would argue that the two problems even interact. Generally Bayesian methods should be able to account for the uncertainty, by exploring the posterior distribution of β and by that different rules and boundaries. This is not possible with the convergence problems in the case of separation. If the boundary can be captured with a rule and this rule separates the data, it will have a unreasonable high β coefficient and the model is unable to explore other rules. What might happen is, that the prediction of cases which for which a seperating rule is true, will be solely based on this rule, no matter of its values in other covariates. This is not good, as intuitively it makes sense to look at other covariates if an observation is a close call. Therefore the overestimation of certainty of regions directly translates into a deterioration of predictive performance.
An exception is the Application on the Prostate Cancer data set, where the Hs-RuleFit showed excellent performance. One explanation is that the main difficulty in gene expression data is to find the right predictors. The convergence problems do not effect the sensitivity of the Horseshoe prior, that is the ability to detect the strong signals. On the contrary, the strong signals are potentially overestimated, which seems to be a minor concern in this context.
A conclusion could be that when fitting Rule Ensembles one might not seek to regularize as drastically to account for the uncertainty about the measurement of y. As we did not observe the underlying latent variable z we should be cautious about a found rule, even if it seems to fit perfectly on the seen examples. Instead it might be a good idea to keep more competing rules than necessary in the model, especially when the number of observations is small. There might be possibilities to incorporate this idea into the horseshoe framework, but it was outside the scope of this work.
Another way to get more gradual boundaries would be to use "soft rules" instead of the hard thresholding rules. "Soft Rules" for Rule Ensembles are used in [2]. The idea is to use the logistic function, as done in the prostate cancer application. Instead of $r_k(x) \in \{0, 1\}$ soft rules take values on the interval between 0 and 1, which can be interpreted as the probability that a rule is true for an observation. Observations close to the decision boundary of the rule will have lower values. This captures the concept of close cases better, with which the rule ensemble models seems to struggle and might help to get more realistic estimation of uncertainty.

5.2 Method

Rule Generation and Hypotheses The empirical results imply that variety in the Tree Ensembles helps the model to find good rules. Using a mixture of Random Forest and Boosting generated rules decreased the prediction error in regression. Utilizing mixtures of Trees is novel and is not used in the literature. To justify I would like to follow up with the interpretation of rule ensembles as expert opinions. The Hs-RuleFit generates a set of competing hypotheses about the data which afterwards get evaluated - very critically - by the horseshoe regularization procedure. It is important to note that the evaluation of the hypotheses is always in context to all other hypotheses in the set, meaning the same rule can be either used or removed depending on the remaining rules. An ideal would be to search for the set of hypotheses which would be chosen even after seeing all other possible hypotheses, the globally best rules. Mixing different types of generating processes is a step in this direction, as it creates a higher variety of hypotheses *against* which the rules are evaluated.
The terms hypotheses and Hypotheses testing are used very lose here, but in my opinion this interpretation makes sense, due to the special shape of the horseshoe prior. A rule gets either rejected or is taken into the model in full magnitude, while intermediate values are unlikely a priori. This naturally gives an analogous to the testing and rejecting of hypotheses [15][20]. The RuleFit framework offers an efficient way to create a variety of hypotheses while the horseshoe prior offers a strong testing procedure. The Lasso does not allow this interpretation, as it artificially overshrinks informative rules and takes uninformative rules

to substitute for the shrinkage as discussed in Section 2.2.
For future work in my opinion especially three directions would be interesting:

1. Randomizing the Tree Generating parameters further, in the same manner as the treedepth. That is for each tree also vary the minimum number of observations in the nodes, the resampling strategy or the loss function used in a random manner. The options are manifold. Inducing more randomness into the Tree generating process would create rule ensembles of even higher variety.

2. In this work only Rules generated from decision Trees were used. In theory any type of learning method that partitions the data in some way could be utilized, with the only constraint being the interpretability. An interesting candidate are neurons from artificial neural networks. Neural networks showed to be strong in finding highly non-linear but natural subspaces, for example faces in image recognition. If such a natural interpretation of a neuron is possible it would be a great addition for the rule ensemble, as this highly non-linear but smooth relationships are hard to approximate via decision rules.

3. An interesting option could be to include user defined hypotheses in the ensemble to see if the data verifies certain hypotheses in the light of alternative hypotheses produced by the Hs-RuleFit. This could be interesting for example in social science to empirically test hypotheses derived from theories.

MCMC-Sampling This study used Gibbs-Sampling to obtain posterior samples for the horseshoe regression. Gibbs-Sampling is the easiest to implement and showed to be computationally feasible for the examined problems. In the regression case the mixing and convergence of the MCMC-chain was very good, however it showed problems in the case of classification. This work mainly tried to improve the mixing through alternative prior specifications, which lower the tails of the half-Cauchy distribution, with moderate success. A different approach could be to use alternative Sampling schemes especially the Hamiltonian MCMC-Sampler [49]. The Hamiltonian could lead to big improvements through its ability to jump between the modes as done in [42]. My theoretical expectation is that the predictive performance of Hs-RuleFit depends crucially on the ability to visit the different modes of the posterior of $\boldsymbol{\beta}$. The problems in classification can in my opinion partly attributed to the mixing problems. More efficient sampling schemes which do not get stuck in one mode for a long time are therefore highly desirable and should be tested in future work.

Horseshoe Prior for classification The theoretical properties of the Horseshoe prior in conjunction with binary regression are understudied, but [53] note that the half-Cauchy distribution as a prior for the scale of β might not be appropriate here. Problems seem especially sincere when the posterior distribution has many modes, as the bad mixing hurts the ability to visit the different modes. Potentially running the MCMC to obtain samples in the area of millions could allow the sampler to visit the different modes, however this is not feasible for medium scale problems. From my current point of view other shrinkage priors with slightly lighter than Cauchy tails could be a better choice for doing classification. The great theoretical properties of the horseshoe prior in regression arise from the infinite probability spike at $\kappa = 0$ which allows to keep signals unshrunk. It is also exactly this spike that causes the problems for classification. Thus within the Horseshoe prior there seems to be a direct conflict in allowing the predictors to be large and at the same time prohibiting the predictors to become unreasonable large in classification. As a consequence the performance of the Hs-RuleFit is worse for classification.
Changing the hyperparamter settings discussed in Section 3.3. helped to improve performance especially on smaller data sets, but it is unclear how it effects the functioning of the horseshoe prior. It has to be expected that the robust prior overshrinks predictors which do

not separate the data, as a side effect. In classification under the horseshoe seems to be a trade-off between the overshrinking of effects and ensuring decent mixing properties.
An interesting alternative to the horseshoe prior is the double-Pareto prior by Armagan et. al.[5]. The implied prior distribution on β under the double-Pareto prior has a tail heaviness in between the exponential distribution and the Cauchy distribution. This could be highly beneficial, as it might prohibit too large coefficients, without harming its ability to detect signals. The double-Pareto especially allows to chose hyper-parameters to manipulate the tail heaviness directly. It also would be interesting to use a prior which puts less confidence on single decision rules and encourages the model to explore more different rule configurations. Therefore a more informative shrinkage prior than the horseshoe might be necessary in classification.
In general more research is necessary about the theoretical properties of shrinkage priors, especially for classification. As noted by Bhattacharya et. al. [9] practitioners operate in the dark, when using shrinkage priors. Shrinkage priors show impressive empirical results, but are not well understood yet. Further theoretical knowledge could help to bypass the problems encountered in this project.

Interpretability This work used mainly visualization tools to interpret the Hs-RuleFit. Especially the novel RuleHeat was useful to evaluate which role a decision rule plays in the ensemble. Looking at the rules which define a group of interest gives an easy way to extract information from the Hs-RuleFit. Friedman & Popescu [27] use partial dependence plots for interpreting the RuleFit. Partial dependence plots allow to analyze how changes in the covariates effect the predictions of the whole ensemble. In a way this is more correct, as it takes into account all non-zero terms of the ensemble, not only the most important ones. On the other hand it completely disregards the special decision rule structure, as partial dependence plots can be as well used for black box models. The RuleHeat was only used on the ten most important decision rules. Therefore it does not reflect the behavior of the whole ensemble, but it grants an insight in the most important mechanisms in the data by exploiting the decision rule structure. In my opinion the Hs-RuleFit promises to be a great data exploration model. An interesting characteristic of decision trees and decision rules is that they are able to approximate any kind of function. Even though the approximation is often unnatural, for example in the case of linear effects, the rule ensemble is likely to find the interesting subspaces. This subspaces can then be inspected to form new hypotheses about the data. Returning to the question if the horseshoe prior leads to interpretable models, I would answer with a clear yes. The produced set of rules are compact enough to identify the important decision rules and give a good insight in the data.

Posterior Summaries The DSS-posterior summary used in this paper is helpful, but it would be interesting to explore other options. It might be possible to choose some of the posterior modes, which cover the most posterior density, directly. This is closer to the discrete-mixture literature, which usually choose a number of models with high marginal posterior probability. Adapting this approach would allow to utilize the vast amount of literature about model selection in the discrete-mixture case. Exploring the modes directly could not be done in this work and might be challenging, but would be interesting for further research, both in the context of Hs-RuleFit but also for other shrinkage prior applications, where finding good posterior summaries is still an open question.

Computational Limitations For small to medium scale data sets (up to $N = 5000$) the Hs-RuleFit implementation was quite feasible. Running it on the Boston Housing data set ($N = 506$) takes around 5 minutes. For Regression the computational complexity depends mostly on the dimensionality of the ruleset and only linearly on the number of observations. Therefore extensions to big scale data sets are possible, especially if the sampling procedure

is emigrated in a faster programming language such as C++. Classification is more expensive due to the Data augmentation, but still feasible for medium sized data sets. Interestingly for very high dimensional data, like genetic expression data the Hs-RuleFit works quite efficient, as the sampling procedure uses only the decision rules, not the original high dimensional data.

5.3 Implications for other areas

Machine Learning In my opinion the here presented results imply that the horseshoe prior and shrinkage priors in general could very well be applied in other machine learning areas. Through the careful choice of prior distributions the horseshoe prior very clear cut achieves what it is used for, in this case identifying very informative decision rules and removing the rest. Each area of machine learning has potentially different requirements on the regularization procedure. An example is the here used incorporation of the rule structure as prior information. This makes Bayesian shrinkage priors a good choice, as it allows to model this requirements explicitly. With the Lasso this is not straightforward, if not impossible. At the same time shrinkage priors are highly adaptive, making it a good default choice for many data situations. This is an advantage over discrete-mixture priors which are generally quite dependent on the choice of suitable hyper parameter settings. Also shrinkage priors are computationally more efficient, which is important in high dimensional settings. With this the Bayesian shrinkage priors appear to be well suited for a broader field of regularization problems in machine learning.

Science and daily life The results presented in this work promise, that highly accurate and yet simple understandable models are indeed possible. This has implications for both science and daily life. If models are simple enough, that it does not need a statistician to understand them, they could be used for example in personalized medicine, in form of cellphone applications. Already many people collect data about their health and try to find pattern, which is an difficult task. An interpretable algorithm could assist in that.
The here proposed algorithm could be especially interesting in social sciences and medicine. The algorithm is well suited to be used as an exploratory tool for big data sets, which become more and more common. The interpretational form gives the option to compare a model with existing theories and form novel hypotheses. In a sense interpretable classifiers could form a bridge between machine learning and the very theory driven social sciences. They are much more accurate than the traditionally used linear models and at the same time allow interpretation, which is a requirement in this area.
Another interesting area of usage of the Hs-RuleFit and alike is artificial intelligence and robotics. As pointed out before decision rules are easy interpretable for humans, but even more so for computer systems. The simple logical form of the resulting model could be easily integrated in a larger system consisting of perception, predictions and actions. The same way the Hs-RuleFit can guide further human research it could be used to guide a robots interaction with its environment.

6 Conclusion

I have proposed Hs-RuleFit a new model for regression and classification. Hs-RuleFit showed great empirical performance in regression. Together with novel visualization tools it produced easy interpretable models, which have predictive accuracy on par or better than the popular methods Random Forest, BART and RuleFit. Therefore the application of the horseshoe prior on rule ensemble regression was successful. The empirical performance suggest that Hs-RuleFit is a good choice in regression problems.
The success in the regression setting did not immediately carry over to classification where problems occurred. Future work should focus on the classification aspect of shrinkage priors which is still understudied, in order to make shrinkage priors applicable here. I think if a shrinkage prior is designed to explicitly meet the requirements of classification as described above, shrinkage priors could work well here as well. The good results in regression motivate that it is worth the effort to investigate this line of research further.
Shrinkage priors showed a flexibility which could make it also a great choice for other machine learning methods, like neural networks and Support Vector machines. In this work it was possible to incorporate prior information about the decision rule structure and by that adapt the regularization procedure to prefer simple and easy interpretable rules. As a consequence Hs-RuleFit is less prone to overfitting, which was one weakness of the original RuleFit. Due to its flexibility and its adaptivity the horseshoe prior showed in this work superior performance over the Lasso. The connection of machine learning and Bayesian shrinkage priors is very promising and deserves further investigation.

Appendix A

The full-conditional distribution for $\nu_i|\cdot$ differs from the one presented by Makalic & Schmidt [44].
Using the decomposition $\lambda \sim \mathcal{C}^+(0, \eta)$ as inverse gammas leads to

$$\lambda^2|\nu \sim \mathcal{IG}(0.5, \nu^{-1}), \tag{6.1}$$
$$\nu \sim \mathcal{IG}(0.5, \eta^{-2}), \tag{6.2}$$

with the densities

$$f(\lambda^2|\nu) = \frac{\nu^{-0.5}}{\Gamma(0.5)}\lambda^{-1.5}exp(-\nu^{-1}\lambda^{-2}), \tag{6.3}$$
$$f(\nu) = \frac{\eta^{-1}}{\Gamma(0.5)}\nu^{-1.5}exp(-\nu^{-1}\eta^{-2}). \tag{6.4}$$

Using the Chainrule gives

$$f(\lambda^2, \nu) = f(\lambda^2|\nu)f(\nu) \tag{6.5}$$
$$= \frac{\eta^{-1}}{\pi}\lambda^{-1.5}\nu^{-1-1}exp(-\nu^{-1}(\lambda^{-2} + \eta^{-2})) \tag{6.6}$$
$$= \frac{\eta^{-1}}{\pi}\lambda^{-1.5}\frac{1}{\lambda^{-2}+\eta^{-2}}\underbrace{(\lambda^{-2} + \eta^{-2})\nu^{-1-1}exp(-\nu^{-1}(\lambda^{-2} + \eta^{-2}))}_{\mathcal{IG}(1,\lambda^{-2}+\eta^{-2})} \tag{6.7}$$

Thus $f(\lambda^2) = \frac{\eta^{-1}}{\pi}\lambda^{-1.5}\frac{1}{\lambda^{-2}+eta^{-2}}$ and $\nu|\lambda^2, \eta^2 \sim \mathcal{IG}(1, \frac{1}{\lambda^2} + \frac{1}{\eta^2})$. Then using the same Decomposition on $\eta \in C^+(0, A)$, which leads to

$$\eta^2|\psi \sim \mathcal{IG}(0.5, \psi^{-1}), \tag{6.8}$$
$$\psi \sim \mathcal{IG}(0.5, A^{-2}). \tag{6.9}$$

Again using the Chain rule leads to

$$f(\psi|\eta^2) = \frac{1}{\pi}\eta^{-1.5}A^{2^{-1.5}}\frac{1}{\eta^{-2} - A^{-2}}\underbrace{(\eta^{-2} + A^{-2})\psi^{-1-1}exp(\frac{1}{\psi}(\eta^{-2} + A^{-2})}_{\psi|\eta^2\sim\mathcal{IG}(1,\eta^{-2}+A^{-2})} \tag{6.10}$$

and

$$f(\nu, \psi, \eta^2) = f(\nu|\eta, \psi) f(\eta|\psi) f(\psi) \tag{6.11}$$

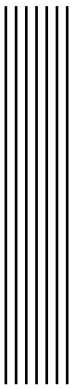
$$= \underbrace{\frac{1}{\pi} \nu^{-1.5} \psi^{-0.5} \frac{1}{\nu^{-1} + \psi^{-1}}}_{f(\psi,\nu)} \underbrace{(\nu^{-1} + \psi^{-1}) \eta^{-1-1} exp(\frac{1}{\eta}(\nu^{-1} + \psi^{-1}))}_{\eta|\psi,\nu \sim \mathcal{IG}(1, \nu^{-1}+\psi^{-1})}. \tag{6.12}$$

The full-conditionals required for sampling are:

$$\nu|\lambda^2, \eta^2 \sim \mathcal{IG}(1, \frac{1}{\lambda^2} + \frac{1}{\eta^2}), \tag{6.13}$$

$$\eta|\psi, \nu \sim \mathcal{IG}(1, \frac{1}{\nu} + \frac{1}{\psi}), \tag{6.14}$$

$$\psi|\eta^2 \sim \mathcal{IG}(1, \frac{1}{\eta^2} + \frac{1}{A^2}). \tag{6.15}$$



Bibliography

[1] Timo Aho et al. "Multi-target regression with rule ensembles". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 2367–2407.

[2] Deniz Akdemir, Nicolas Heslot, and Jean-Luc Jannink. "Soft Rule Ensembles for Supervised Learning". In: *stat* 1050 (2013), p. 22.

[3] Adelin Albert and JA Anderson. "On the existence of maximum likelihood estimates in logistic regression models". In: *Biometrika* 71.1 (1984), pp. 1–10.

[4] James H Albert and Siddhartha Chib. "Bayesian analysis of binary and polychotomous response data". In: *Journal of the American statistical Association* 88.422 (1993), pp. 669–679.

[5] Artin Armagan, David B Dunson, and Jaeyong Lee. "Generalized double Pareto shrinkage". In: *Statistica Sinica* 23.1 (2013), p. 119.

[6] Maria Maddalena Barbieri and James O Berger. "Optimal predictive model selection". In: *Annals of Statistics* (2004), pp. 870–897.

[7] Anindya Bhadra et al. "The horseshoe+ estimator of ultra-sparse signals". In: *arXiv preprint arXiv:1502.00560* (2015).

[8] Anirban Bhattacharya, Antik Chakraborty, and Bani K Mallick. "Fast sampling with Gaussian scale-mixture priors in high-dimensional regression". In: *arXiv preprint arXiv:1506.04778* (2015).

[9] Anirban Bhattacharya et al. "Bayesian shrinkage". In: *arXiv preprint arXiv:1212.6088* (2012).

[10] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[11] Leo Breiman. "Bias, variance, and arcing classifiers". In: (1996).

[12] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[13] Leo Breiman et al. *Classification and regression trees*. CRC press, 1984.

[14] Carlos M Carvalho, Nicholas G Polson, and James G Scott. "Handling sparsity via the horseshoe". In: *International Conference on Artificial Intelligence and Statistics*. 2009, pp. 73–80.

[15] Carlos M Carvalho, Nicholas G Polson, and James G Scott. "The horseshoe estimator for sparse signals". In: *Biometrika* (2010), asq017.

[16] Saptarshi Chakraborty and Kshitij Khare. "Convergence properties of Gibbs samplers for Bayesian probit regression with proper priors". In: *arXiv preprint arXiv:1602.08558* (2016).

[17] Hugh A Chipman, Edward I George, and Robert E McCulloch. "BART: Bayesian additive regression trees". In: *The Annals of Applied Statistics* (2010), pp. 266–298.

[18] William W Cohen. "Fast effective rule induction". In: *Proceedings of the twelfth international conference on machine learning*. 1995, pp. 115–123.

[19] William W Cohen and Yoram Singer. "A simple, fast, and effective rule learner". In: *Proceedings of the national conference on artificial intelligence*. JOHN WILEY & SONS LTD. 1999, pp. 335–342.

[20] Jyotishka Datta, Jayanta K Ghosh, et al. "Asymptotic properties of Bayes risk for the horseshoe prior". In: *Bayesian Analysis* 8.1 (2013), pp. 111–132.

[21] Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. "ENDER: a statistical framework for boosting decision rules". In: *Data Mining and Knowledge Discovery* 21.1 (2010), pp. 52–90.

[22] Manuel Fernández-Delgado et al. "Do we need hundreds of classifiers to solve real world classification problems?" In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3133–3181.

[23] Yoav Freund, Robert E Schapire, et al. "Experiments with a new boosting algorithm". In: *ICML*. Vol. 96. 1996, pp. 148–156.

[24] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[25] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[26] Jerome H Friedman and Bogdan E Popescu. "Importance sampled learning ensembles". In: *Journal of Machine Learning Research* 94305 (2003).

[27] Jerome H Friedman and Bogdan E Popescu. "Predictive learning via rule ensembles". In: *The Annals of Applied Statistics* (2008), pp. 916–954.

[28] Johannes Fürnkranz. "Separate-and-conquer rule learning". In: *Artificial Intelligence Review* 13.1 (1999), pp. 3–54.

[29] Andrew Gelman et al. "A weakly informative default prior distribution for logistic and other regression models". In: *The Annals of Applied Statistics* (2008), pp. 1360–1383.

[30] Andrew Gelman et al. "Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper)". In: *Bayesian analysis* 1.3 (2006), pp. 515–534.

[31] Edward I George and Robert E McCulloch. "Variable selection via Gibbs sampling". In: *Journal of the American Statistical Association* 88.423 (1993), pp. 881–889.

[32] Joyee Ghosh, Yingbo Li, and Robin Mitra. "On the use of Cauchy prior distributions for Bayesian logistic regression". In: *arXiv preprint arXiv:1507.07170* (2015).

[33] Enrico Glaab et al. "Using rule-based machine learning for candidate disease gene prioritization and sample classification of cancer gene expression data". In: *PloS one* 7.7 (2012), e39932.

[34] Jim E Griffin, Philip J Brown, et al. "Inference with normal-gamma prior distributions in regression problems". In: *Bayesian Analysis* 5.1 (2010), pp. 171–188.

[35] P Richard Hahn and Carlos M Carvalho. "Decoupling shrinkage and selection in Bayesian linear models: a posterior summary perspective". In: *Journal of the American Statistical Association* 110.509 (2015), pp. 435–448.

[36] Chris Hans. "Bayesian lasso regression". In: *Biometrika* 96.4 (2009), pp. 835–845.

[37] Rie Johnson and Tong Zhang. "Learning nonlinear functions using regularized greedy forest". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36.5 (2014), pp. 942–954.

[38] Hyunjoong Kim et al. "Visualizable and interpretable regression models with good prediction power". In: *IIE Transactions* 39.6 (2007), pp. 565–579.

[39] Max Kuhn and Kjell Johnson. *Applied predictive modeling*. Springer, 2013.

[40] James Ladyman. *Understanding philosophy of science*. Psychology Press, 2002.

[41] Benjamin Letham et al. "An interpretable stroke prediction model using rules and Bayesian analysis". In: (2013).

[42] Longhai Li and Weixin Yao. "Fully Bayesian Logistic Regression with Hyper-Lasso Priors for High-dimensional Feature Selection". In: *arXiv preprint arXiv:1405.3319* (2014).

[43] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: `http://archive.ics.uci.edu/ml`.

[44] Enes Makalic and Daniel F Schmidt. "A simple sampler for the horseshoe estimator". In: *arXiv preprint arXiv:1508.03884* (2015).

[45] Nicolai Meinshausen. "Node harvest". In: *The Annals of Applied Statistics* (2010), pp. 2049–2072.

[46] Juan C Meza. "A numerical comparison of rule ensemble methods and support vector machines". In: *Lawrence Berkeley National Laboratory* (2010).

[47] Michael Molla et al. "Using machine learning to design and interpret gene-expression microarrays". In: *AI Magazine* 25.1 (2004), p. 23.

[48] Radford M Neal. "Slice sampling". In: *Annals of statistics* (2003), pp. 705–741.

[49] Radford M Neal et al. "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo* 2 (2011), pp. 113–162.

[50] Trevor Park and George Casella. "The bayesian lasso". In: *Journal of the American Statistical Association* 103.482 (2008), pp. 681–686.

[51] Nicholas G Polson and James G Scott. "Shrink globally, act locally: Sparse Bayesian regularization and prediction". In: *Bayesian Statistics* 9 (2010), pp. 501–538.

[52] Nicholas G Polson, James G Scott, and Jesse Windle. "Bayesian inference for logistic models using Pólya–Gamma latent variables". In: *Journal of the American statistical Association* 108.504 (2013), pp. 1339–1349.

[53] Nicholas G Polson, James G Scott, et al. "On the half-Cauchy prior for a global scale parameter". In: *Bayesian Analysis* 7.4 (2012), pp. 887–902.

[54] Adrian E Raftery, David Madigan, and Jennifer A Hoeting. "Bayesian model averaging for linear regression models". In: *Journal of the American Statistical Association* 92.437 (1997), pp. 179–191.

[55] Lior Rokach. "Ensemble-based classifiers". In: *Artificial Intelligence Review* 33.1-2 (2010), pp. 1–39.

[56] Håvard Rue. "Fast sampling of Gaussian Markov random fields". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 325–338.

[57] Robert E Schapire. "A brief introduction to boosting". In: *Ijcai*. Vol. 99. 1999, pp. 1401–1406.

[58] Robert E Schapire et al. "Boosting the margin: A new explanation for the effectiveness of voting methods". In: *Annals of statistics* (1998), pp. 1651–1686.

[59] Dinesh Singh et al. “Gene expression correlates of clinical prostate cancer behavior”. In: *Cancer cell* 1.2 (2002), pp. 203–209.

[60] Alan Sokal. “What is science and why should we care”. In: *Vortrag, gehalten am* 27 (2008), p. 2008.

[61] Carolin Strobl, James Malley, and Gerhard Tutz. “An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests.” In: *Psychological methods* 14.4 (2009), p. 323.

[62] Aik Choon Tan and David Gilbert. “Ensemble machine learning on gene expression data for cancer classification”. In: (2003).

[63] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[64] Satyanshu K Upadhyay et al. *Current Trends in Bayesian Methodology with Applications*. CRC Press, 2015.

[65] Peng Zhao and Bin Yu. “On model selection consistency of Lasso”. In: *The Journal of Machine Learning Research* 7 (2006), pp. 2541–2563.

[66] Christopher Zorn. “A solution to separation in binary response models”. In: *Political Analysis* 13.2 (2005), pp. 157–170.

ISRN: LIU-IDA/STAT-A--16/009--SE