Optimizing Rule Selection in Random Forest Simplification

Bachelor thesis by Timo Henz
Date of submission: January 29, 2021

1. Review: Eneldo Loza Mencía
2. Review: Michael Rapp
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Knowledge Engineering Group

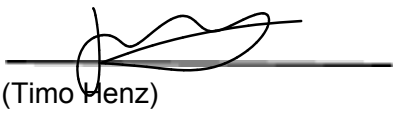
Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Henz Timo, die vorliegende Bachelor-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, den 29.01.2021



(Timo Henz)

Abstract

State of the art machine learning methods for classification such as random forests provide high accuracy and robust results. However, their outputs are hard for humans to interpret. This work introduces a method to strategically select a subset of rules, called a simplified random forest, from a random forest. This work introduces a method to strategically select a subset of rules from a random forest, called a simplified random forest. This achieves a higher interpretability while preserving the best classification possible with fewer rules. In the process several methods of evaluating simplified random forests are explored.

The results achieved by this method show that it is possible to achieve accuracies comparable to the whole random forest with very few rules and even to outperform the random forest with a subset of its rules. The classification often outperforms baseline strategies for selecting subsets of rules. However, it is inconsistent which loss function performs the best and the margin to baseline approaches is not very broad.

Contents

1 Introduction

Machine learning is a powerful technology that can assist or replace human decision making in all kinds of fields including medicine, business economics, mechanical engineering, human resources or social media.

Besides optimizing the quality of classifications such as increasing accuracy for making better and more reliable decisions, recent developments in the field shows “the need for moving from simply accepting what a model does, to interpreting it to understand how it does so” (Silva, Freitas, and Handschuh, 2019). This is important to ensure safety (Otte, 2013), prevent discrimination and comply with laws.

Many prediction models reflect and amplify social biases found in data (Zhao, Wang, Yatskar, Ordonez, and Chang, 2017). Consequently systems can also become the source of discrimination e.g. when in 2015 Google Photos accidentally labeled people of color as gorillas (Simonite). Understanding what a system decides might also be a legal requirement such as the European Parliament’s regulation on data protection and privacy General Data Protection Regulation (GDPR) which effectively creates “the right to non-discrimination and the right to explanation” (Goodman and Flaxman, 2017).

Interpretability is hard to define - especially with an approach that is supposed to be generically applicable to any use case. An interpretable answer needs “the ability to explain or to present in understandable terms to a human” (Doshi-Velez and Kim, 2017). When creating a generic applicable approach it is hard to consider the technical relationships between statements. Especially whether certain termini imply or contradict each other.

Statements that are understandable by humans (at least by experts in their respective fields) are often represented by so called rules. Under the premise that fewer rules are easier to understand for humans, which can be derived from Doshi-Velez and Kim (2017) hypothesis, the goal is to find preferably small prediction models.

In this work a method called "gain covering" is proposed. It relies on continually assessing the benefits of every additional rule in a sequential rule selection process.

1.1 Simplifying Random Forests

This work builds on an machine learning technique called random forests which aggregates many decision trees to achieve better and more robust results. Decision trees are a machine learning model from which rules can be extracted. The exact process will be explained later.

Preceding works show that random forests for classification can be simplified by extracting only a few rules from a random forest (Rapp, Loza Mencía, and Fürnkranz, 2019). This increases interpretability while potentially maintaining a good accuracy. While this necessarily implies a trade-off, it is evident that a cleverly selected set of rules can maximize the accuracy that is reachable with the limited amount of rules.

This approach is supposed to be generically applicable to any random forest. Random forest can widely vary in size. Use cases for real world applications vary a lot as well: Depending on the specific use case, its need of interpretability and quality of classification, the adequate trade-off is context dependent.

Therefore it is necessary to be able to extract a simplified random forest of any size out of the original random forest of any size. An assessment whether a specific size has still sensible accuracy of classification is important.

1.2 Evaluating Results

Another important challenge is finding out how to measure the quality of predictions provided by a subset of rules. A measurement is needed that takes all sizes of simplified random forests (from a subset with one rule to a subset of n rules) into account. It is a principal goal to improve interpretability of the results by reducing the number of rules. Therefore, the measurement needs to reward good results with very few rules. It should be applicable to any random forest and also take varying random forest sizes into account. With this measurement that is applicable to any random forest it is possible to compare all results from this new approach to existing baseline.

1.3 Contributions

Thus, this work makes two achievements: it proposes a method for strategically selecting rules for a simplified random forest that works with loss functions and it introduces measurements for evaluating methods that select rules for simplified random forests.

The goal for the method proposed in this work is to find small models for higher interpretability while still preserving the highest accuracy possible. The best trade-off possible should be achieved by finding an optimal loss function or the best loss function for a specific situation.

A secondary goal of this method is to find out whether it is possible to consistently surpass the accuracy of the underlying random forest.

The goal for the measurements is to assess the quality of classification a method can provide that is applicable to random forests of all sizes and all methods.

2 Fundamentals

This first chapter gives an introduction to binary classification and a few central terms in this field. Afterward an overview over baseline techniques is given.

2.1 Classification

Classification is a process in which an instance or a set of instances are assigned to distinct categories. The decision to which of these categories an instance belongs is made based on its characteristics and features. When there are only two categories the task is called binary classification where one category is the positive class and the other one the negative class. Analogous to statistical hypothesis testing a correct classification is called true and an incorrect classification false.

The results of such a classification can be represented in a confusion matrix. Correct classifications of the positive class are called true positives, correct classifications of a negative example are called true negatives. An incorrect negative prediction of a positive class is called false negative. An incorrect positive prediction for a negative class is called false positive.

		Actual class	
		P	N
Predicted class	P	TP	FP
	N	FN	TN

2.2 Test Data and Training Data

Supervised machine learning are machine learning techniques, where a model is trained on a set of training instances and evaluated on a set of test instances (to measure how good the classification is). This distinction between trainingdata and testdata is important. The risk of not using test- and trainingdata is overfitting of trainingdata and thus not knowing how good the classification is for unseen data. Overfitting is a common problem in machine learning where the algorithm keeps improving on trainingdata while its classification is stagnating or getting worse on testdata.

Furthermore, any process for building simplified random forests and evaluating them will occur several times with different distinct trainingdata and testdata in a process called "cross-validation". In cross validation it is not necessary to have several times the amount of instances to train and test with. Instead the union set of trainingdata and testdata gets dissected several times into different pairs of distinct testdata and trainingdata. Cross validation is a useful tool for the credibility of results in a proof-of-concept and makes substantiated statements about them more plausible.

2.3 Ensemble Methods

Ensemble methods, such as bagging, are techniques in machine learning where many weak learners are combined to create one strong learner. From the perspective of the bagging algorithm every individual learner is a black box that makes a decision - in this case a binary categorization. Notably, weak learners neither have to be very elaborate nor achieve highly accurate results on their own - indeed they can be quite naive. The bagging algorithm takes all decisions by all weak learners into account to make one overall decision.

Hereby the different ensemble algorithms take several strategies including majority vote, weighted voting, sequential votes etc. Advantageous for producing a good correctness and robust results is a high number of overall weak learners and how well they cover training- and test-instances together.

Two broad categories of ensemble methods are boosting and bagging. The main difference is that bagging algorithms use weak learners that are independent of each other and the bagging makes a decision based on all of these weak learners (Breiman, 1996) while

boosting algorithms work sequentially - every weak learner builds on its predecessor and corrects previous errors (Freund, Schapire, and Abe, 1999). A more elaborated comparison between bagging and boosting has been done by Maclin and Opitz (1997).

2.4 Random Forests

The foundational work for the following is a random forest. A random forest is a bagging technique that uses a homogeneous set of weak learners called decision trees (Breiman, 2001). In a decision tree the instances are classified by passing them through several condition-tests, which are placed on the nodes of a tree model. The branches of the node represent the outcomes of the decision test. Starting with the root on every node the instance continues on the branch where the condition is fulfilled. The leaves represent individual classes. In the case of binary classification the leaves are thus 1 or 0. When an instance gets classified by a decision tree the instance is passed only through all conditions from the root to the leaf where the instance ultimately ends - other conditions from nodes that the instance did not pass have no impact on this classification.

Figure 2.1 gives an example of how decision trees can be represented. This visualization is reminiscent of flow charts, which is a intuitive way to understand them and one of the reasons why they are a popular way to get foundational data from humans in real world applications.

The random forest takes all decision trees into account. The advantage of this technique emerges especially when trees complement each other or address different cases.

2.5 Rule extraction

To get a more fine grained control over the ensemble process individual rules get extracted from decision trees. A rule is the the set of conditions extracted from its decision tree. It contains all conditions from one leaf up to the root. Thus, there are as many rules as there are leaves from all decision trees in the random forest combined.

In this work, a rule is defined as consisting of a head and a body. The head represents the class that gets selected when the rule is fulfilled. The body is a conjunction of conditions. An instance fulfills a rule if all conditions on this instance are true and the instance gets classified as the class in the head of the rule.

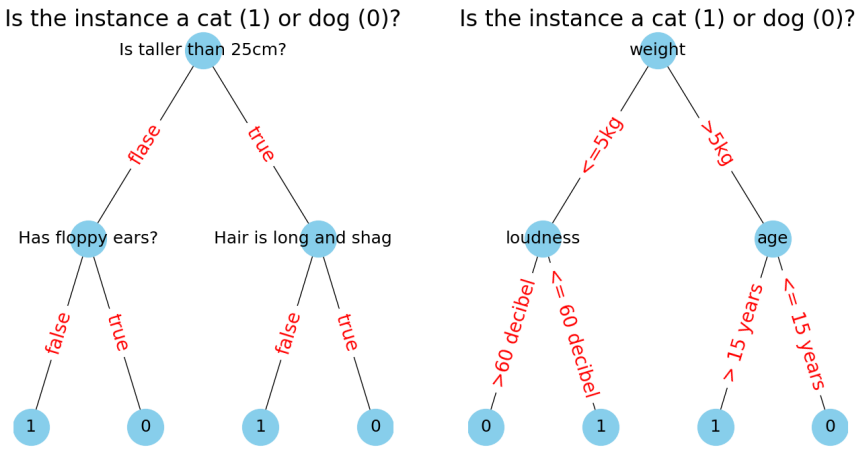


Figure 2.1: Visual representation of two decision trees. In this example both decision trees do not provide compelling classification on their own.

So, an instance gets classified as true by a rule when this rule is derived from the leaf the instance would finish the decision tree on. Correspondingly an instance also fulfills all conditions in a path in the decision tree to this leaf and doesn't on any other route.

Figure 2.2 gives one example of a rule extraction from a decision tree and shows the boolean condition to fulfill the rule.

In the following different methods will be discussed to take advantage of the this more fine grained control over the bagging process with individual rules instead of whole decision trees.

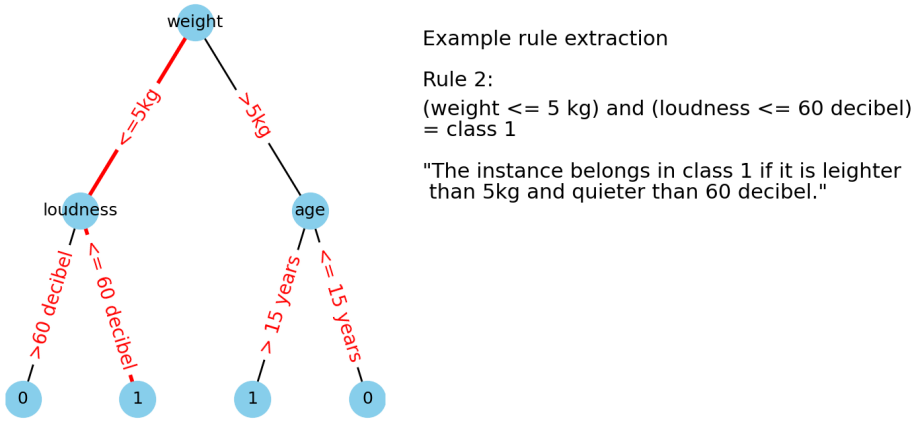


Figure 2.2: One rule consists of several conditions from the decision tree.

3 Baseline Methods

The following chapter introduces baseline approaches derived from Rapp et al. (2019). The method explored in later chapters will be measured against these methods by trying to achieve better results or by deriving measurements from these baseline methods.

In the following all methods start with set of all rules extracted from its random forest. Duplicate rules got removed beforehand. The goal is building sets of rules of all sizes between only one rule and all rules available. For every size the accuracy is measured with testdata (which does not impact the rule selection process). Every subset of rules is a simplification of a random forest.

3.1 Exploration Space

When selecting a subset of rules there are many possible combinations. When searching for the best combination it is a straight forward idea to get all combinations and brute force through them. However, an exhaustive search is limited by the size of the exploration space. The number of possible combinations when selecting r out of n rules (disregarding ordering which is irrelevant here) is

$$C(n,r) = \binom{n}{r} = \frac{n!}{(n-r)!(r!)}$$

One Random Forest tested for this paper is hearth-statlog.arff with 16651 rules. Selecting e.g. 5 Rules out of 16651:

$$C(16651,5) = \frac{16651!}{(16651-5)!(5!)} = 1.066 * 10^{19}$$

For real world applications with eventually larger Random Forests and even with state of the art hardware it is an unrealistic scenario to try all of them. First experiments with this

baseline idea of brute forcing through a few million combinations showed pretty quickly that due to overfitting trainingdata this approach cannot keep up with baseline approaches presented in [simplifying random forests: on the trade-off between interpretability and accuracy] and will not be used in the following.

3.2 Random Rules

The first baseline approach is selecting rules randomly. By definition the results will vary widely. For comparison other approaches get compared to averaged results of multiple random rule selections.

Selecting random rules is not only a baseline approach but also a good indicator for measuring results and whether strategic planning is effective at all.

3.3 Best Rules

This baseline approach sorts all rules by a heuristic from best rule to worst rule.

The most intuitive to understand this idea is with the heuristic "Correctly Classified" which counts the number of true positives and true negative: The method lets all rules classify every instance of trainingdata. The heuristic compares these classifications with the correct classification of the trainingdata instances and sums up the score of correct classifications. The rule with more correct classifications come before rules with less correct classifications. Noteworthy heuristics are accuracy, precision, recall and m-estimate.

In the following the terms true positive (TP), false positive (FP), true negative (FN) and false negative (FN) are abbreviated.

The accuracy heuristic classifies by calculating the number of correct predictions (true positives and true negative) divided by the number of total instances to classify. The accuracy is thus always between 1 and 0 or expressed by a percentage.

$$Accuracy = \frac{(TP + FN)}{(TP + TN + FP + FN)}$$

The precision heuristic is the proportion of true positives among the number of instances classified as positive. Thus, precision heavily penalizes false negatives.

$$Precision = \frac{(TP)}{(TP + FP)}$$

The recall heuristic is the proportion of true positives among the number of positive instances. It can be expressed as the percentage of correctly classified positive instances. Recall is oblivious to the classification of false instances.

$$Recall = \frac{(TP)}{(TP + FN)}$$

The m-estimate heuristic is the implementation of a maximum-likelihood estimation. It is the precision value plus the proportion of true instances among all instances multiplied by an m-factor. The default value for m is 22.466. According to Rapp et al. the best results can be achieved with m-estimate.

$$M - estimate = \frac{TP + m * ((TP + FN)/(TP + TN + FP + FN))}{(TP + FN) + m}$$

Every subset of n rules is then the first n rules of this sorted list.

The key disadvantage of selecting the best rules first seems to be that many good rules cover the main broad lump set of instances and fail to cover more niche/rare cases. Thus, one of the key benefits of ensemble methods is restricted: rules that compliment each other are not deliberately selected.

3.4 Weighted Covering

Weighted Covering is the only baseline approach which selects rules one by one sequentially and tries to avoid redundant coverage of the same instances.

Firstly, all instances from the training dataset get assigned a weight of 1. After selecting a rule all instances that are covered by this rule get their weight cut in half. Every iteration all rules (that were not selected yet) accumulate the weights of the instances that they would cover. The rule with the highest accumulated weight is selected next. With every iteration the weights fall. After the broad lump set of instances declines in weight, the more niche and unique rules become worth it to get covered. Thus, weighted covering actively selects rules that compliment each other.

Nevertheless the rules that work good together are selected only indirectly over the coverage of instances. The risk is misjudging the importance of niche or rare instances: the rules that cover these more niche rules could be harmful for the overall result for other instances. weighted covering is oblivious to this risks because it never measures the confusion matrices with all instances.

4 Related Work

This chapter gives a short insight in other methods to increase the interpretability of machine learning models and the groundwork about makes a machine learning model interpretable.

The idea to modify large machine learning models and extract smaller parts of it to boost classification accuracy or increase interpretability is not new. Rapp et al. (2019) demonstrates this by introducing the rule extraction and creating simplified random forests with the baseline approaches random rules, best rules and weighted covering.

An open question in the field is what makes a machine learning system interpretable or not interpretable. This question won't get answered here. A relevant factor for this has been hypothesized by Doshi-Velez and Kim (2017) as the form of cognitive chunks, number of cognitive chunks, level of compositionality, monotonicity and other interactions between cognitive chunks and uncertainty and stochasticity. When designing a generic solution for random forests the form of cognitive chunks, level of compositionality and uncertainty and stochasticity depend on the input random forest. To manipulate the monotonicity and other interactions between cognitive chunks or level of compositionality in random forests it would be necessary to look into the rules - this idea does not get pursued here as well. simplifying random forests mainly improves the number of cognitive chunks by straightforwardly reducing the number of rules. Of course, Doshi-Velez and Kim (2017) hypothesized other metrics for interpretability as well.

Another paper exploring possibilities to interpret artificial intelligence systems is Silva et al. (2019) with systematic strategies such as distributional semantics and fuzzy logic.

In Vellido, Martín-Guerrero, and Lisboa (2012) several ideas are discussed to explain artificial intelligence decision making from the special session on interpretable models in machine learning, organized as part of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.

Of course, there are other works which study interpretability with machine learning models more similar such as decision trees or specifically random forests.

In Souad and Abdelkader (2019) a random forest gets pruned while trying to retain diversity. For this an entropy function is used as a fitness function for a genetic algorithm. This is very similar in a sense that it takes interdependency of different decision trees into account. The fundamental difference is that Souad and Abdelkader (2019) starts a new processing step after a finished random forest, instead of beginning with a new boosting process from the ground up. Furthermore the decision trees do not get dissected into rules.

Hatwell, Gaber, and Azad (2020) introduced a machine learning model called CHIRPS that dissects decision trees form random forests into paths (equivalent to the rules used in this work) and builds a rule via frequent pattern mining from these paths by selecting specific conditions. This manipulates the underlying random forest in a much more fundamental way than the explored methods in this work.

Another approach in this field by Quinlan (1987) is to simplify decision trees. This method wasn't applied to a whole random forest. However, if applied to all trees in a random forest it wouldn't take interdependency of trees into account.

4.1 Research Gap

None of the methods specifically about random forest start a new bagging process from the bottom up but build a process on top of random forests. Another unique property about the process presented in this work is that the basic building blocks are individual rules instead of whole decision trees. This presents a fundamentally more fine grained perspective. Despite this difference the diversity based entropy function used in Souad and Abdelkader (2019) is similar to loss functions. The entropy function does not explicitly measure correct and incorrect classifications - it measures overall statistical diversity instead of a boosting-inspired sequential loss function. The CHIRPS method provided by Hatwell et al. (2020) does work with individual rules from random forests but extracts the most commonly occurring split conditions which is a different perspective from looking at outcomes of classifications and using the rules for a new bagging process.

5 Methodology

This chapter explains the rule selection of the here presented method, called gain covering. The following section defines the procedure of every iteration to add an additional rule to the model.

The process begins with a gain of zero and after selecting a rule to add to the model, the gain of the selected rule gets subtracted from the current gain.

5.1 Gain Covering

This new approach selects rules sequentially based on their helpfulness on training data. For this all rules (that have not been selected yet) iterate through all training instances I.

Figure 5.1 shows the iteration through all rules that have not been selected yet as the input parameter *listOfRemainingRules* for the method *nextIteration* which represents the selection of one rule. As every rule itarates through every instance the *nextIteration* method figure 5.1 has a nested loop through all instances *listOfInstances* inside the loop through all remaining rules.

The contender rule r_c makes a classification with the previous selected rules (resulting in a contender model m_c) for every Instance $i_1 ... i_m$ in the trainingdata. Every contender model saves the number of correct and incorrect classifications in a confusion matrix counting its true positive, false positives, false negative and true negative predictions. The sum of true positive and true negative (correct classifications) minus the sum of false positive and false negative (incorrect classifications) is called the margin, which is an important input for a lot of loss functions.

Figure 5.1 presents the process of rules voting on the respective instance, the counting of correct and incorrect classifications and the result from the loss function in the blackbox

method *lossfunction* which needs all previously selected rules as *lastModel*, the contender rule as *contenderRule* and the respective instance.

The resulting values from every contender model get accumulated over all instances. The contender model with the highest sum of loss function results and is the next model m.

In figure 5.1 the sum of the loss function results is in the *additionalGain* variable. The best rule (which is the additional rule in the contender model with the highest sum) is variable *bestRule* and the gain it would provide in the *bestgain* variable.

In the first iteration this highest sum is called gain. In every following iteration the gain is the difference between the previous gain and the highest sum of loss function results. So the gain from this iteration gets subtracted from the previous gain (or zero if it is the first iteration) to calculate the resulting gain for the next iteration.

The pseudocode in figure 5.1 has the previous gain as input parameter *lastGain* and calculates the gain for the next iteration in the end as *nextGain*.

5.1.1 Adjusting the Ensemble Method

This algorithm picks up element from bagging and boosting. The rule selection in a sequential manner is reminiscent of boosting while the final simplified random forest still classifies like a bagging algorithm. The differences of boosting and bagging are mentioned in Chapter 2.3.

The key advantage of the loss functions in this step is that they introduce an adjustable component which specifies the strategy in the rule selection process. Thus, they are modular interchangeable pieces that can be tailored to the specific requirements which the simplified random forest should fulfill.

Loss functions can also do an implicit seperate and conquer strategy such as weighted covering if they penalize redundant instance coverage.

After looking at the gain covering process, the next chapter shows several loss functions which stand for different strategies in the rule selection.

```
#Method to calculate the next selected rule
#@param lastGain The achieved gain from previous iterations or 0 in first iteration
#@param lastModel Set of rules that have already been selected in previous iterations
#@param listOfRemainingRules Set of rules that have not been selected yet
nextIteration(lastGain, lastModel, listOfRemainingRules){
        bestgain = -Infinity;
        bestRule;
        for(rule: listOfRemainingRules){
                additionalGain = 0;
                for(instance: listOfInstances){
                        additionalGain += lossfunction(lastModel,contenderRule,instance);
                }
                if(additionalGain > bestgain){
                        bestgain = additionalGain;
                        bestRule = rule;
                }
        }
        nextGain = lastGain-bestgain;
        nextModel = lastModel.add(bestRule);
        nextRemainingRules = listOfRemainingRules.remove(bestRule)

        return(nextGain, nextModel, nextRemainingRules)
}


#Method to calculate the result of the loss function
#if lastModel and contenderRule did the next classification together
#@param lastModel Set of rules that have already been selected in previous iterations
#@param contenderRule Rule that gets added to the lastModel non-permanently
#@param instance that gets classified by all rules from lastModel and contenderRule
        which leads to the number of correct/incorrect classifications which serves as
        the input for the loss function
lossfunction(lastModel,contenderRule,instance){}
```

Figure 5.1: Pseudocode for one rule selection.

5.2 Loss Functions

After looking at the process how gain covering sequentially selects rules with summed up results from a loss function, this section introduces a few loss functions and visualizes their different outputs.

Chapter 5.1 explains the context in which the loss function is used. Throughout the rule selection process, every contender rule makes classifications about one instance with the rules that are already in the model from previous iterations. Together, they vote on this instance. They either cover this instance or they do not cover the instance. If they cover the instance, the classification can either be correct or it can be incorrect.

All loss functions use this information how many correct and incorrect classifications the rules from this contender model made. They are used as input for the loss functions as can presented in table 5.1. Δ represents the difference between incorrect and correct votes. Ψ is the sum of incorrect and correct votes, it does not contain the rules that do not cover the instance. Θ is the number of possible votes if all rules classified and covered the instance.

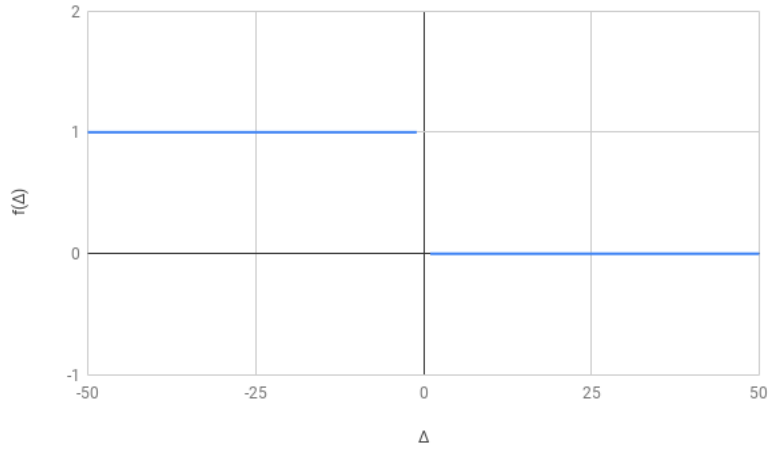
All loss functions in the following sections have mathematical function defined and a graphic that represents how a higher δ or a more negative δ or a δ closer to 0 impacts the result of the loss function. Section 5.3 explains what that means for the relation of rewarding good results and penalizing bad results.

Symbol	Definition
Δ	number of correct votes - number of incorrect votes
Θ	number of possible votes
Ψ	number of correct votes + number of incorrect votes

Table 5.1: The used input parameters derived from.

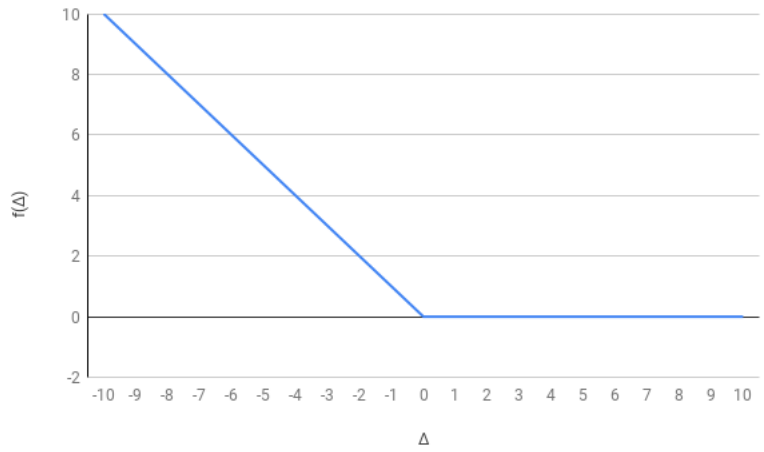
5.2.1 Zero-one Loss

$$\textit{zero-one},\ f(\Delta) = \begin{cases} 1 & \text{if } \Delta < 0 \\ 0 & \text{if } \Delta >= 0 \end{cases}$$



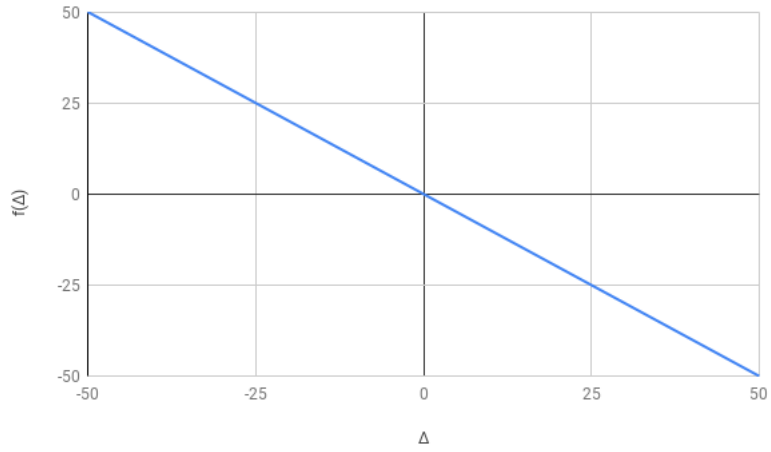
5.2.2 Hingeloss-classic

$$\textit{hingeloss-classic},\ f(\Delta) = \begin{cases} -\Delta & \text{if } \Delta < 0 \\ 0 & \text{if } \Delta >= 0 \end{cases}$$



5.2.3 Absolute margin Loss

absolutemarginloss, $f(\Delta) = -\Delta$

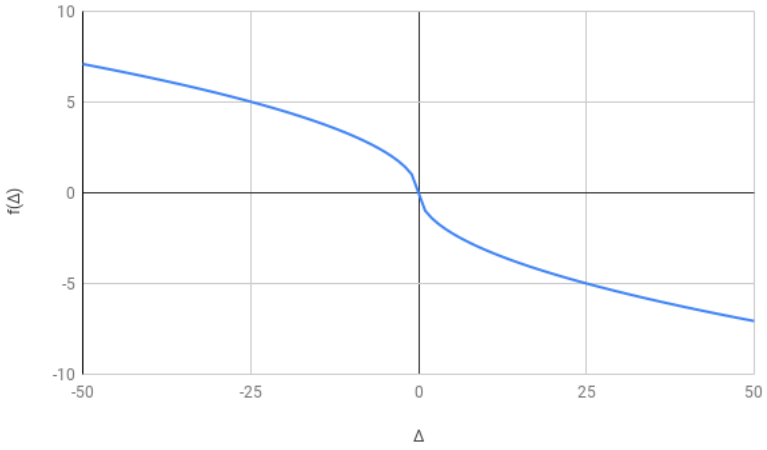


5.2.4 Relative margin Loss

relativemarginloss, $f(\Delta) = -\Delta/\Theta$

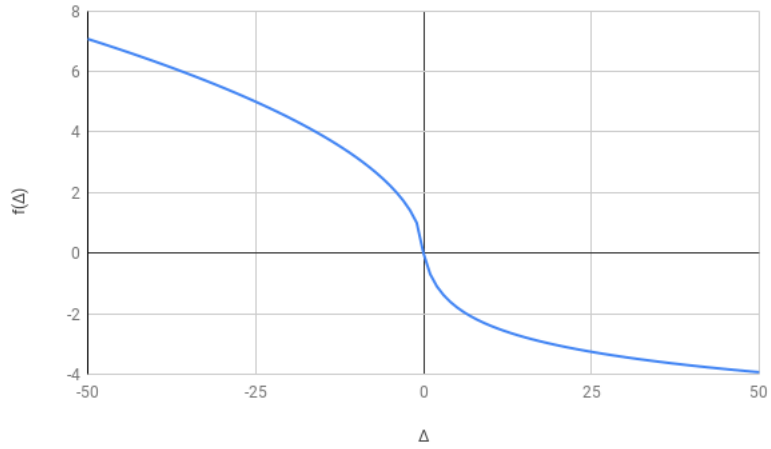
5.2.5 Absolute Root Error

$$absoluterooterror,\ f(\Delta) = \begin{cases} \sqrt{|\Delta|} & \text{if } \Delta < 0 \\ 0 & \text{if } \Delta = 0 \\ -\sqrt{|\Delta|} & \text{if } \Delta > 0 \end{cases}$$



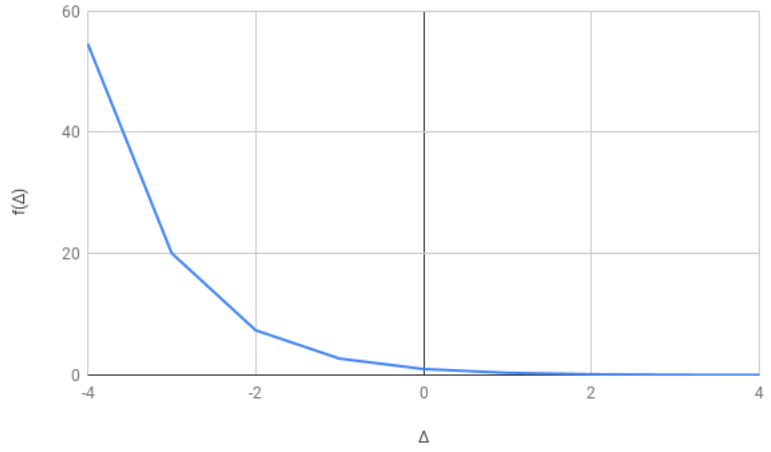
5.2.6 Asymmetric

$$asymmetric,\ f(\Delta) = \begin{cases} \sqrt{|\Delta|} & \text{if } \Delta < 0 \\ -\ln(\Delta + 1) & \text{if } \Delta >= 0 \end{cases}$$



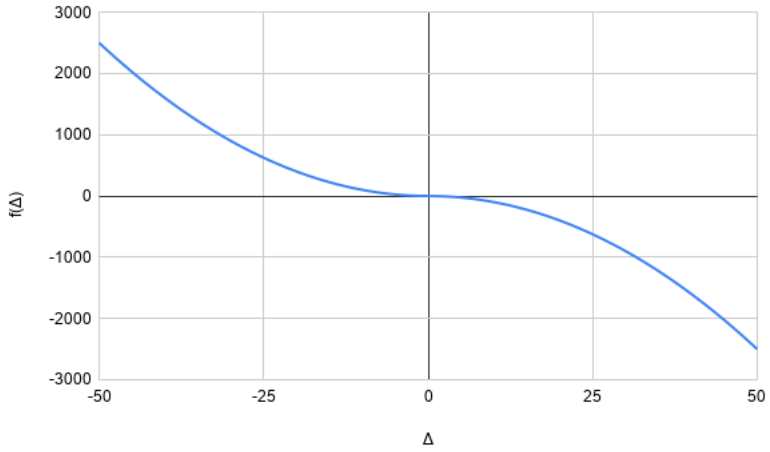
5.2.7 Exponential

exponential, $f(\Delta) = e^{-\Delta}$



5.2.8 Absolute Square Margin

$$\textit{absolutesquaremargin},\ f(\Delta) = \begin{cases} \Delta^2 & \text{if } \Delta < 0 \\ 0 & \text{if } \Delta = 0 \\ -\Delta^2 & \text{if } \Delta > 0 \end{cases}$$



5.2.9 Logistic Loss

$\textit{logistic},\ f(\Delta) = log_2(\Theta + e^{-\Delta})$

5.3 Comparisons

Among these loss function are relatively simple ideas such as zero-one, hingeloss-classic and absolutemarginloss. To understand how these loss functions affect the gain of every classification it is important to think about how the right side represents rewarding correct classification and the left side represents penalizing incorrect classification.

Notable about differences of loss functions is that e.g. hingeloss-classic has blanket $+1$ penalty for all overall worse classification while e.g. absolutemarginloss continuously rewards and penalizes improvements and declines in classification.

Similarly, there are big differences in the more complex loss functions. Absoluterooterror and absolutesquaremargin are opposites in how they handle increasing rewards and penalties. Absolutesquaremargin has increasingly growing rewards for better Δs while absoluterooterror has stagnantly growing rewards. Thus, it is an open question which of these strategies achieve the better accuracies. Another loss function relevant for this comparison is asymmetric which is similar to absoluterooterror with emphasized rewards.

The linear fashion in which absolutemarginloss and relativemarginloss have rewards and penalties that are in between absolutesquaremargin and absoluterooterror.

5.4 Example Rule Selection

For a better understanding of the way gain covering uses a loss function to select a specific example subset of rules this section explains the algorithm based on an example. This example is artificial - normally the random forest contains more rules and more instances are needed to for reliable results.

The goal of this example is to select a subset of 3 out of 4 rules resulting in model m_3. For this, the correct (true positive and true negative) and incorrect (false negative and false positive) classifications. The following table 5.3 shows how the four example rules classify the four example instances. in this example all rules cover all instances. This is not the case in the datasets and real world examples where rules do not always cover an instance in which case they simply wouldn't contribute to the δ as explained in section 5.2.

Parameter	Symbol	Value
All rules	R	r_1, r_2, r_3, r_4
Number of rules	$\|R\|$	4
Goal subset of rules		m_3
Trainingsinstances	I	i_1, i_2, i_3, i_4
Number of instances	$\|I\|$	4
Loss function	$f()$	absoluterooterror

Table 5.2: Used parameters and goals for this example.

	r_1	r_2	r_3	r_4
i_1	1	-1	1	-1
i_2	1	1	-1	1
i_3	-1	1	-1	-1
i_4	1	-1	-1	-1

Table 5.3: overview which loss functions cover which instances correct (1) or incorrect (-1). Correct includes true positive and true negative. Incorrect includes false positive and false negative.

5.4.1 First Iteration

The model has no rules selected yet. Contender model m_{0+r_1} consists of rule r_1 only. It classifies instance i_1 correctly which means one correct vote and zero incorrect votes (there are no other Rules in m_{0+r_1}).

$$\Delta = 1, f(\Delta) = \begin{cases} \sqrt{|\Delta|} & \text{if } \Delta < 0 \\ 0 & \text{if } \Delta = 0 \\ -\sqrt{|\Delta|} & \text{if } \Delta > 0 \end{cases}$$

$$f(1) = -1$$

The results of the loss function of m_{0+r_1} for the other instances are $-1; 1; -1$ respectively. The accumulated result for m_{0+r_1} over all instances is $-1 - 1 + 1 - 1 = -2$. The potential gain from this contender model is thus -2, since this is the first iteration and the gain begins with 0. The other rules are tried identically in their contender models.

	m_{0+r_1}	m_{0+r_2}	m_{0+r_3}	m_{0+r_4}
gain	$0-(-2)$ $= 2$	$0-(0)$ $= 0$	$0-(2)$ $= -2$	$0-(0)$ $= -2$

The first model is $m_1 = \{r_1\}$.

5.4.2 Second Iteration

The contender models in the second iteration are m_{1+r_2}, m_{1+r_3} and m_{1+r_4}. They consist of r_1 and the respective rule that has not been selected yet.

Contender model m_{1+r_2} classifies i_1 one time correct and one time incorrect. This can be seen in table 5.3 where r_1 (which is already in the model from the previous iteration) classifies i_1 correctly and r_2 classifies i_1 incorrectly. Since δ is the difference between correct votes and incorrect votes, the resulting δ is 0.

$$\Delta = 1 - 1 = 0, f(\Delta) = \begin{cases} \sqrt{|\Delta|} & \text{if } \Delta < 0 \\ 0 & \text{if } \Delta = 0 \\ -\sqrt{|\Delta|} & \text{if } \Delta > 0 \end{cases}$$

$$f(0) = 0$$

m_{1+r_2} classifies i_2 correctly twice.

$$\Delta = 2, f(2) = \begin{cases} \sqrt{|2|} & \text{if } 2 < 0 \\ 0 & \text{if } 2 = 0 \approx -1.41 \\ -\sqrt{|2|} & \text{if } 2 > 0 \end{cases}$$

m_{1+r_2} classifies i_3 one time correct and one time incorrect.

$$\Delta = 0, f(0) = 0$$

m_{1+r_2} classifies i_4 one time correct and one time incorrect.

$$\Delta = 0, f(0) = 0$$

The results of the loss function of m_{1+r_2} for the all instances are $0; -1.41; 0; 0$. The accumulated result for m_{0+r_1} over all instances is $0 - 1.14 + 0 + 0 = -1.41$. The gain from from the previous iteration and this contender model result in a gain of $2 - (-1.14) = 3.41$ After calculating the same for the the other two rules the results are:

	m_{1+r_2}	m_{1+r_3}	m_{1+r_4}
gain	$\approx 2-(-1.41)$ $= 3.41$	$\approx 2-(0)$ $= 2$	$\approx 2-(0)$ $= 2$

The next model is $m_2 = \{r_1, r_2\}$.

5.4.3 Final Iteration

The contender models are m_{2+r_3} and m_{2+r_4} and the achieved gain is 4.41.

Contender model m_{2+r_3} classifies i_1 and i_2 correctly with $\delta = 1$. It classifies i_3 and i_4 incorrectly with $\delta = -1$. After applying loss functions the summed up gain is $1+1-1-1 = 0$.

Contender model m_{2+r_4} classifies i_2 correctly with $\delta = 1$. It classifies i_1, i_3 and i_4 incorrectly with $\delta = -1$. After applying loss functions the summed up gain is $-1+\sqrt{3}-1-1 \approx -1.27$.

	m_{2+r_3}	m_{2+r_4}
gain	$\approx 3.41-(0)$ $= 3.41$	$\approx 3.41-(-1.27)$ $= 4.68$

The final model is $m_2 = \{r_1, r_2, r_4\}$.

6 Evaluating Iterative Selection of Rules

This chapter shows the results from simplified random forests from different datasets and different loss functions.

6.1 Measurement by Area under Curve

The first evaluation is based on the idea that the necessary simplification and thus the necessary simplified random forest size is unknown (dependent on context of the specific domain).

By measuring the accuracy of all simplified random forests, from a simplified random forest with one rule, a simplified random forest with two rules to a simplified random forest with all rules, the progess of the rule selection is measured explicitly. When representing this process in a graph with the number of rules in the respective simplified random forest on the x axis and the achieved accuracy on the y axis the subsequent curve represents the progess of the rule selection. The integral of this curve, which represents the area under the curve, can be calculated by the accumulated sum of all accuracies.

Figure 6.1 visualizes the area under curve and the accuracy missing for the maximum achievable result which would be achieved when every single simplified random forest reached an accuracy of 100. The dark area is the achieved result by the integral and the light blue area is what is missing for the maximum achievable result. The area under the curve can thus be represented as a percentage.

Because of the final integral score being an artificial figure 7.1 shows the results ranked. When several loss functions achieved the same result their scores over their consecutive ranks are averaged. The last column represents the summed up ranks.

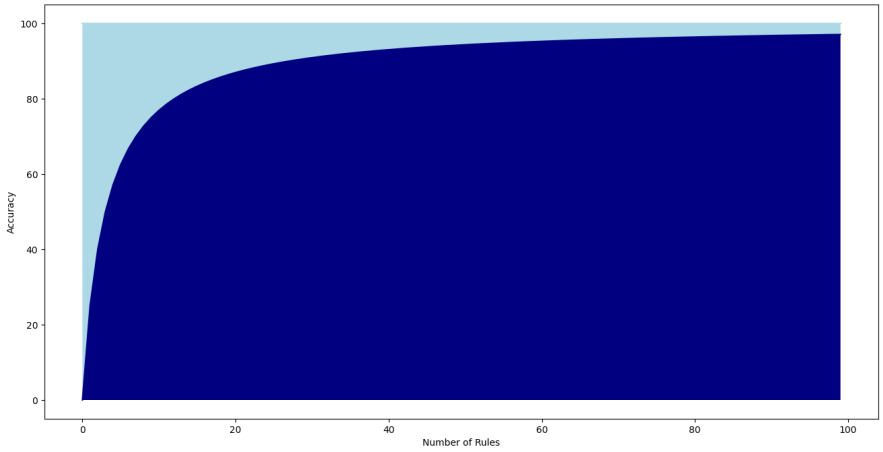


Figure 6.1: Visual representation of the area under curve (dark) and the accuracy missing for the maximum achievable result (light blue). The dark are is thus a representation of the achieved percentage of the maximum integral score.

6.1.1 Weight Function

One major problem with examining the integral is that it does not directly reward fast climbing accuracy. Especially since the curve is not always monotonically increasing the loss function could have received great results with few rules, fall afterwards and thus receive a bad score with the integral despite its good results with few rules.

To get around this problem a weight function is introduced that makes simplified random forests with fewer rules more important than simplified random forests with more rules.

To implement such a weight function every achieved accuracy gets multiplied with a factor between 1 and 0.

6.1.2 Discounted Cumulative Gain

Discounted cumulative gain (dcg) is a technique used by web search engine algorithms with a similar use case: it measures the usefulness, of a document based on its position in the result list.

$$f(x) = \frac{1}{log(x+1)/log(2)},$$

x = number of rules in simplified random forests, f(x) = weight factor

Figure 6.2 visualizes the area under curve with the discounted cumulative gain as a weight function and the accuracy missing for the maximum achievable result which would be achieved when every single simplified random forest reached an accuracy of 100. Notable about this weight function is that the weight falls quite sharply in the beginning. The simplified random forest with three rules has only a weight of 0.5 already. The difference in weight falls more slowly in the later rules.

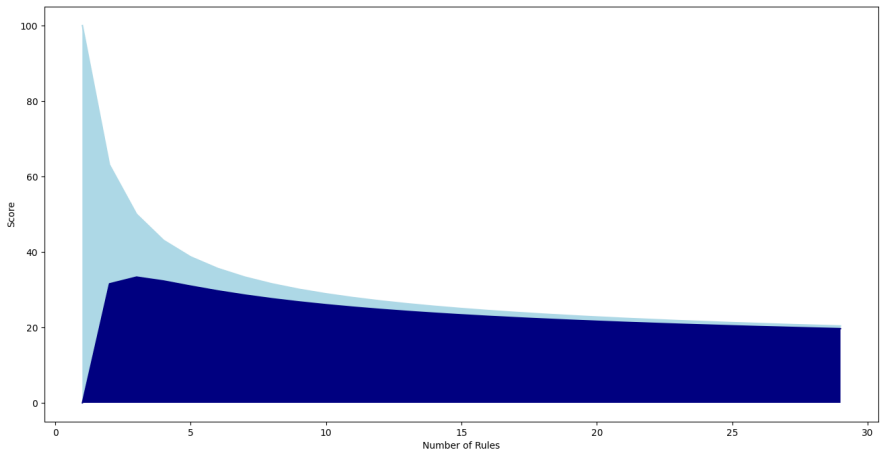


Figure 6.2: Visual representation of the area under curve (dark) and the accuracy missing for the maximum achievable result (light blue) with a discounted cumulative gain function. The achieved score is derived from the accuracy and the weight function and represent as the share of the area under the light blue curve.

6.1.3 Linear Weight Function

Another possible weight function is a linear weight function. The advantage of this linear weight function is that the weight for the first few simplified random forests don't fall quite

as fast. Additionally the weights continue to fall when the number of rules approaches the end.

$$f(x) = f(x, n) = \frac{n-x}{n},$$

x = number rules in simplified random forests, n = number of all rules, f(x) = weight factor

Figure 6.3 visualizes the area under curve with the linear weight function and the accuracy missing for the maximum achievable result which would be achieved when every single simplified random forest reached an accuracy of 100.

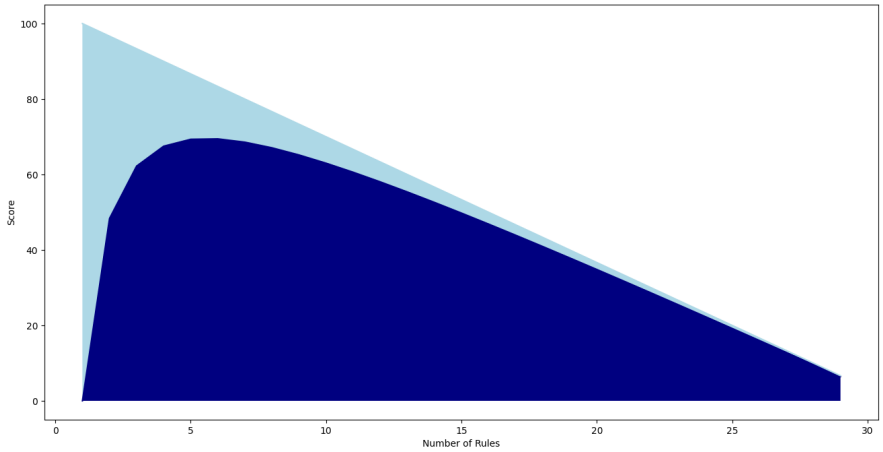


Figure 6.3: Visual representation of the area under curve (dark) and the accuracy missing for the maximum achievable result (light blue) with a linear weight function. The achieved score is derived from the accuracy and the weight function and represent as the share of the area under the light blue curve.

6.2 Measurement by Stopping Criteria

The comparisons based on area under curve show differences of accuracy between loss functions. However, no loss function has turned out to be consistently better than others.

In order to get more specific information about which loss function performs well for a specific requirements the loss functions will be measured by a variety of stopping criteria.

A stopping criterion is a cut-off point in the process of the rule selection and an accuracy measurement of the last simplified random forest. Thus, the result consists of the number of rules and the achieved accuracy. Depending on the specific stopping criterion either a lower number is better or higher is better .

6.2.1 Full Coverage

Since the process of rule selection results in growing simplified random forests and therefore a growing number of instances covered by the simplified random forests. A covered instance, in this case, is an instances that has been classified by at least one rule correctly. The full coverage stopping criterion stops at the first simplified random forest that coveres every training instance.

6.2.2 10 Percent of Rules

The first stopping criterion is setting the number of rules at a fixed number of 10 percent of the whole random forest. At this cut-off point the interpretability should be similar between different loss functions over the same random forest.

6.2.3 Diminishing Returns

The concept of the point of diminishing returns is based on comparing the set of rules that have been selected with the rules that haven't been selected yet.

The rules that have not been selected into the simplified random forest yet can be combined into a simplified random forest in the same way which will be called the counter model in the following. As the number of rules in the simplified random forest rises the number of rules in the counter model falls and thus its accuracy falls as well.

In the process of rule selection there comes a point at which the the selected rules in the simplified random forest overtake the counter model in accuracy. A lower number of rules for this stopping criterion is better and a higher accuracy at the cut-off point is better. This stopping criterion has been tried with testdata and trainingdata.

After this point the rules that have not been selected yet achieve a lower accuracy than the rules already selected. Their benefit is thus limited.

6.3 Breakeven with Random Rules

Preferably, the rule selection should find rules that perform better than random rules. But in the process, the difference to random rules should be shrinking. That means that a strategic rule selection is only worth with less rules than this point.

The breakeven point is a stopping criterion for a comparison to randomly selected rows.

6.4 Surpass Random Forest

When looking at the accuracies achieved it became clear that the results surpass the original random forest quite often. This can be a measurement for comparing loss functions. In a sense this is a stopping criteria as well but it is only focused on maximum accuracy instead of a trade-off that takes interpretability into account.

7 Evaluating Results

This chapter compares the results from the different setups with different measurements.

7.1 Setup

The used datasets included different random forests with instances in .arff files. Table 7.1 gives information about the number of available rules and the the number of instances in the training and test environment in a 5-fold cross-validation.

Dataset	domain	number of rules	number of training instances	number of test instances
breast-cancer	medical	4858	1144	286
heart-statlog	medical	3052	1080	270
hepatitis	medical	2308	620	155
ionosphere	geophysics	1746	1404	351
labor	human resources	1238	228	57
sonar	engineering	1818	832	208
titanic	archaeology	22	8804	2201
vote	politics	3375	1740	435

Table 7.1: Metadata about the used datasets. Datasets are from https://waikato.github.io/weka-wiki/datasets/.

7.2 Evaluation by Area under Curve

The first look on the result of the integral in table 7.2 also gives an idea on how good the classification of each dataset is.

To give a better idea how these numbers compare to each other (and to the two baseline approaches in the last rows) the table in figure 7.1 shows these results ranked and marked with colors.

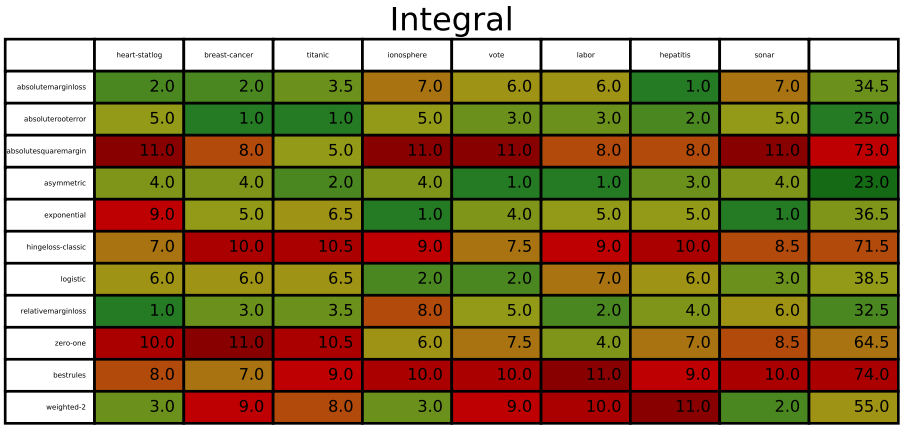


Figure 7.1: Ranked scores over area under the curve and summed up (last column).

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar
hingeloss-classic	0.7661	0.6616	0.4247	0.8774	0.8889	0.7612	0.7562	0.7807
exponential	0.7607	0.6740	0.7534	0.8945	0.8983	0.7979	0.7711	0.8245
logistic	0.7688	0.6731	0.7534	0.8927	0.8987	0.7784	0.7673	0.8092
relativemarginloss	0.7784	0.6837	0.7605	0.8776	0.8949	0.8046	0.7762	0.7835
zero-one	0.7556	0.6615	0.4247	0.8783	0.8889	0.8013	0.7643	0.7807
absoluterooterror	0.7760	0.6843	0.7691	0.8886	0.8985	0.8034	0.7769	0.7967
bestrules	0.7622	0.6704	0.7240	0.8531	0.8776	0.6877	0.7593	0.7618
weighted-2	0.7771	0.6657	0.7305	0.8915	0.8859	0.7413	0.7518	0.8126
absolutesquaremargin	0.6868	0.6683	0.7595	0.8096	0.8050	0.7709	0.7630	0.6760
absolutemarginloss	0.7777	0.6841	0.7605	0.8780	0.8942	0.7951	0.7772	0.7807
asymmetric	0.7763	0.6781	0.7641	0.8910	0.8990	0.8051	0.7763	0.8040

Table 7.2: Achieved score in the integral (the maximum score is 1).

Because the measurements with weight functions produce artificial scores with little real meaning besides comparison between each other figure 7.2 and figure 7.3 only show the scores ranked.

Discounted cumulative gain

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar	
absolutemarginloss	5.0	2.0	3.5	6.0	6.0	5.0	2.0	7.0	36.5
absoluterooterror	3.0	1.0	1.0	5.0	4.0	2.0	1.0	5.0	22.0
absolutesquaremargin	11.0	8.0	5.0	11.0	11.0	8.0	7.0	11.0	72.0
asymmetric	2.0	4.0	2.0	4.0	2.0	1.0	3.0	4.0	22.0
exponential	7.0	5.0	6.5	1.0	3.0	4.0	5.0	1.0	32.5
hingeloss-classic	8.0	10.0	11.0	9.0	8.5	10.0	11.0	8.5	76.0
logistic	6.0	6.0	6.5	2.0	1.0	7.0	6.0	3.0	37.5
relativemarginloss	4.0	3.0	3.5	7.0	5.0	3.0	4.0	6.0	35.5
zero-one	10.0	11.0	10.0	8.0	8.5	6.0	10.0	8.5	72.0
bestrules	9.0	7.0	9.0	10.0	10.0	11.0	8.0	10.0	74.0
weighted-2	1.0	9.0	8.0	3.0	7.0	9.0	9.0	2.0	48.0

Figure 7.2: Ranked scores over area under the curve and summed up (last column).

Linear

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar	
absolutemarginloss	4.0	2.0	3.5	7.0	6.0	5.0	3.0	7.0	37.5
absoluterooterror	2.0	3.0	1.0	5.0	4.0	2.0	1.0	5.0	23.0
absolutesquaremargin	11.0	9.0	5.0	11.0	11.0	9.0	7.0	11.0	74.0
asymmetric	3.0	4.0	2.0	4.0	2.0	1.0	2.0	4.0	22.0
exponential	7.0	5.0	6.5	1.0	3.0	4.0	5.0	1.0	32.5
hingeloss-classic	8.0	10.0	11.0	9.0	7.5	10.0	11.0	8.5	75.0
logistic	6.0	6.0	6.5	2.0	1.0	7.0	6.0	2.0	36.5
relativemarginloss	1.0	1.0	3.5	8.0	5.0	3.0	4.0	6.0	31.5
zero-one	10.0	11.0	10.0	6.0	7.5	6.0	9.0	8.5	68.0
bestrules	9.0	8.0	9.0	10.0	10.0	11.0	8.0	10.0	75.0
weighted-2	5.0	7.0	8.0	3.0	9.0	8.0	10.0	3.0	53.0

Figure 7.3: Ranked scores over area under the curve and summed up (last column).

As expected, the results from a linear weight function differ more from the simple integral than discounted cumulative gain (dcg) differs from the integral. Although the introduction of the weight function did not have profound changes.

The simple loss functions zero-one and hingeloss-classic perform worse when measured with a weight function. These loss functions are also overall relatively bad performing.

Noteworthy about all of these results is the comparison between absoluterooterror and absolutesquaremargin, as they follow contrary strategies like mentioned before. Absoluterooterror performs substantially worse than absolutesquaremargin. This difference is even greater between asymmetric and absolutesquaremargin. Asymmetric is similar to absoluterooterror with emphasized rewards.

With the integral the baseline method bestrules is the worst performing method while the baseline method weighted covering is not outperformed consistently by gain covering with its loss function. Weighted covering even fluctuates between being the best method on the heat-statlog dataset measured with dcg and being on the tail end on most datasets when measured with any weight function.

Ultimately, no loss function or baseline method is consistently the best or consistently the worst with any evaluation by area under curve.

7.3 Stopping Criteria

7.3.1 Full Coverage

The stopping criterion full coverage consists of two parts: The number of rules it takes to cover all instances, which should be preferably low and the accuracy achieved by this point, which should be as high as possible.

As expected the baseline method weighted covering achieves full coverage with relatively few rules. Weighted Covering with cutting weights of covered instances in half is intentionally designed to cover all instances fast, so a good result is to be expected. Likewise the baseline method best rules covers all instances relatively late in the rule selection process, which isn't a surprise either because it doesn't take advantage of the collaboration of rules at all and thus has no factor which would strive towards an overall higher coverage of instances.

Full coverage

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar	
absolutemarginloss	7	5	1	8	7	7	6	7	48.0
absoluterooterror	5	3	1	5	5	4	4	5	32.0
absolutesquaremargin	11	6	5	11	8	8	11	11	71.0
asymmetric	4	2	9	4	4	3	3	4	33.0
exponential	2	10	5	1	1	1	2	2	24.0
hingeloss-classic	10	8	10	6	10	11	8	9	72.0
logistic	2	10	5	1	1	1	1	2	23.0
relativemarginloss	6	4	1	9	6	5	9	6	46.0
zero-one	9	8	10	7	10	9	10	9	72.0
bestrules	8	7	5	10	9	10	7	8	64.0
weighted-2	1	1	1	1	3	6	5	1	19.0

Figure 7.4: Ranked scores over the necessary number of rules to cover all instances with summed up ranks (last column).

Accuracy at the point of full coverage

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar	
absolutemarginloss	3.5	2.0	2.0	6.5	6.0	8.0	1.5	8.0	37.5
absoluterooterror	3.5	3.0	8.0	8.0	2.5	8.0	1.5	2.0	36.5
absolutesquaremargin	11.0	11.0	2.0	11.0	10.0	8.0	4.0	11.0	68.0
asymmetric	2.0	6.0	10.0	5.0	6.0	5.0	7.0	1.0	42.0
exponential	6.0	4.5	10.0	2.5	8.5	6.0	10.0	3.5	51.0
hingeloss-classic	8.5	8.5	4.5	9.0	2.5	4.0	7.0	9.5	53.5
logistic	6.0	4.5	10.0	2.5	8.5	10.0	11.0	3.5	56.0
relativemarginloss	6.0	1.0	2.0	6.5	6.0	2.0	4.0	6.0	33.5
zero-one	10.0	8.5	4.5	4.0	2.5	1.0	4.0	9.5	44.0
bestrules	1.0	7.0	6.0	10.0	11.0	11.0	7.0	5.0	58.0
weighted-2	8.5	10.0	7.0	1.0	2.5	3.0	9.0	7.0	48.0

Figure 7.5: Ranked scores over the achieved accuracy summed up (last column).

These two perspectives on this stopping criterion can sometimes be very different since a bad result in the number of rules can be an advantage for the accuracy at this point and

vice versa for low number of rules and disadvantage for the accuracy. An example of this can be seen in table 7.5 and table 7.4 where logistic loss reaches this point mostly with very few rules but a bad accuracy at this point.

To give a better overview over this connection between the rank in accuracy and the rank in number of rules figure 7.6 shows how different loss functions compare to each other. As a lower rank is better, the best position to be in is the bottom left corner and the worst position is the top right corner. A position further left but higher up on the y-axis means a good rank for the number of rules but a worse rank for the achieved accuracy (e.g. logistic loss). A position further right but lower on the y-axis means a good rank for the accuracy but a worse rank for the number of rules (e.g.relativemarginloss).

Overall the best rules baseline method does not keep up with the results achieved by gain covering while weighted covering performs comparable to other loss functions.

A comparison from the area under curve that still holds true for the full coverage stopping criterion is that absolutesquaremargin performs significantly worse than absoluterooterror and asymmetric.

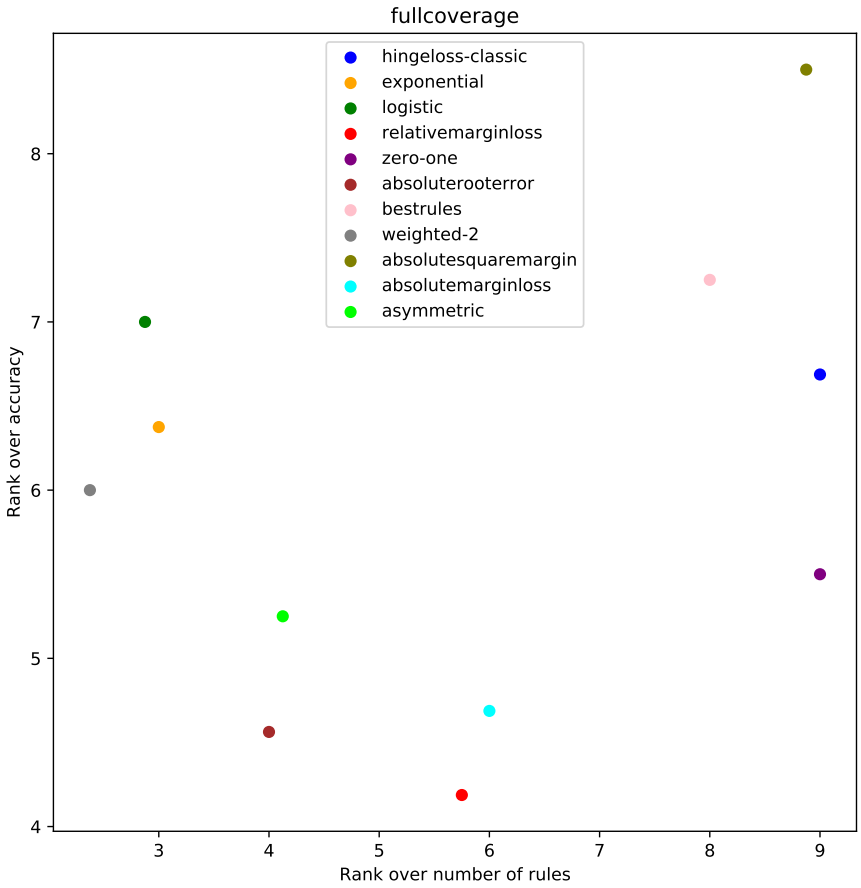


Figure 7.6: Scatter diagram to contextualize the relation between number of rules and accuracy of the full coverage stopping criterion. The ranks are averaged on both axis.

7.3.2 10 Percent of Rules

Table 7.7 shows ranked scores over the achieved accuracy 10 percent of all rules. Similar to the measurements by are under curve the simpler loss functions such as hingeloss-classic and zero-one perform relatively bad. Furthermore the absolutesquaremargin loss function has consistently one of the worst results and performs significantly worse than absoluterooterror and asymmetric here as well. Indeed absoluterooterror and asymmetric are the by and large best loss functions at 10 percent of rules.

Accuracy at 10% of rules

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar	
absolutemarginloss	3.5	3.0	4.0	9.0	5.5	8.0	5.5	7.0	45.5
absoluterooterror	1.0	2.0	4.0	6.0	4.0	2.0	2.0	5.0	26.0
absolutesquaremargin	11.0	8.0	4.0	11.0	11.0	11.0	8.0	11.0	75.0
asymmetric	5.5	4.5	4.0	4.0	1.0	5.0	1.0	4.0	29.0
exponential	8.0	8.0	4.0	1.0	2.5	7.0	3.5	3.0	37.0
hingeloss-classic	9.0	10.5	10.5	7.0	7.5	9.0	11.0	9.5	74.0
logistic	7.0	6.0	4.0	3.0	2.5	3.5	8.0	1.0	35.0
relativemarginloss	3.5	4.5	4.0	8.0	5.5	1.0	5.5	8.0	40.0
zero-one	10.0	10.5	10.5	5.0	7.5	6.0	10.0	9.5	69.0
bestrules	2.0	8.0	9.0	10.0	10.0	10.0	3.5	6.0	58.5
weighted-2	5.5	1.0	8.0	2.0	9.0	3.5	8.0	2.0	39.0

Figure 7.7: Ranked scores over the achieved accuracy at 10% of rules with summed up ranks (last column).

7.3.3 Diminishing Returns

The point of diminishing returns represents the point before which the rules that have not been selected yet provide a higher accuracy when taken together into a counter model than the rules that have been selected. Hence it is better to achieve this point with fewer rules and with a higher accuracy. Consequently the actual number of rules also gives an idea how many rules it takes to get a significant portion of the possible accuracy. Later this idea will be explored further by surpassing the accuracy of the whole random forest. Table 7.3 shows the number of rules at the point of diminishing returns.

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar
absolutemarginloss	117	8	3	253	229	131	114	247
absoluterooterror	38	11	2	299	202	23	11	160
absolutesquaremargin	609	34	3	303	1166	249	110	330
asymmetric	27	11	3	13	262	23	18	109
exponential	7	10	2	23	155	30	248	57
hingeloss-classic	480	13	20	421	973	347	372	364
logistic	7	10	2	23	155	30	190	57
relativemarginloss	114	8	3	256	237	114	113	260
zero-one	851	13	20	1031	973	179	273	364
bestrules	194	9	5	258	405	272	51	295
weighted-2	9	20	5	6	36	126	109	104
number of rules	3052	4858	22	1746	3375	1238	2308	1818

Table 7.3: The number of rules at the point of diminishing returns and the maximum number of rules from the underlying random forest.

As a conclusion from table 7.3, the number of rules necessary to reach the point of diminishing returns is rather low.

Just as full coverage, this stopping criterion can be presented from the perspective of number of rules and from the perspective of the achieved accuracy at this point. A position further left but higher up on the y-axis means a good rank for the number of rules but a worse rank for the achieved accuracy. A position further right but lower on the y-axis means a good rank for the accuracy but a worse rank for the number of rules. The optimal position is again the bottom left corner and the worst position is the top left corner.

Overfitting

A common problem in machine learning is overfitting of training data. It occurs when a machine learning model stops improving on testdata because it adapts to the nuances of the trainingdata to precisely. Normally, ensemble methods such as random forests are very robust against overfitting. Nonetheless, the simplified random forests could be vulnerable to overfitting again due to their less robust coverage over the set of instances and fewer redundant rules.

The easiest way to identify overfitting is finding a correlation between a point after which improvements on trainingdata slow down and and the classification on testdata declines or stagnates. The point of diminishing returns is suited for this comparison because after this number of rules the accuracy left is limited.

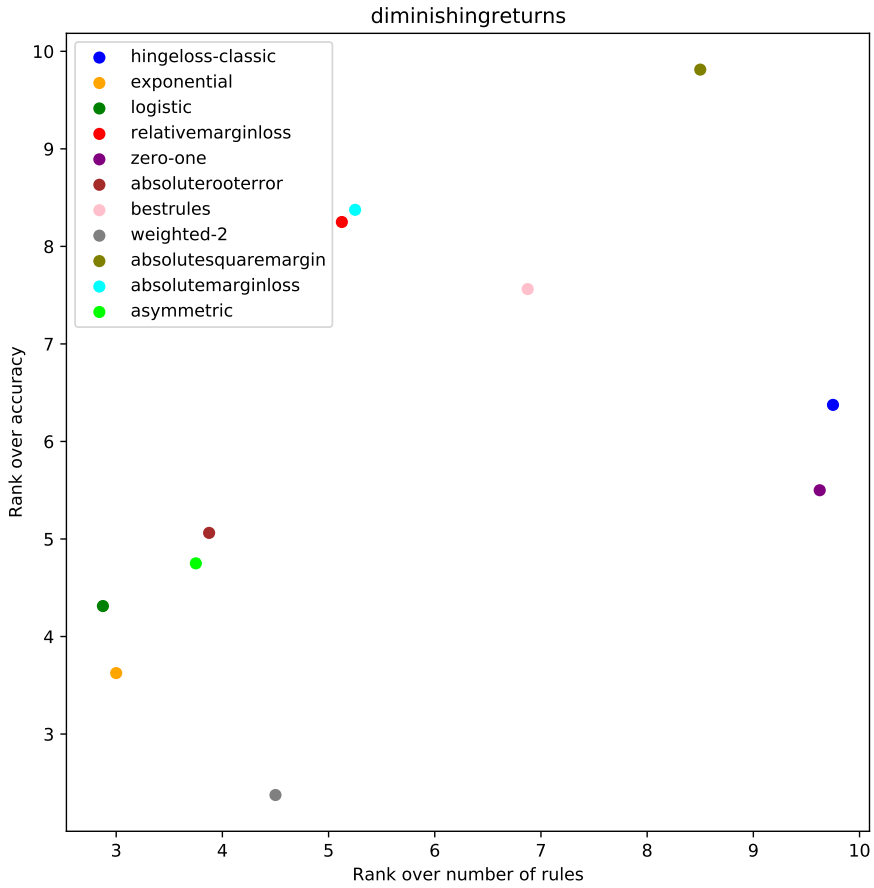


Figure 7.8: Scatter diagram to show the relation between number of rules and accuracy at the stopping criterion. The ranks are averaged on both axis.

The comparison between the diminishing returns on testdata and trainingdata gives insight in the discrepancy between how long it takes to achieve a majority of the achievable accuracy. If a loss function is is further right on the scatter diagram over testdata (see figure 7.8) than on the scatter diagram over training data (see figure 7.9) it means that

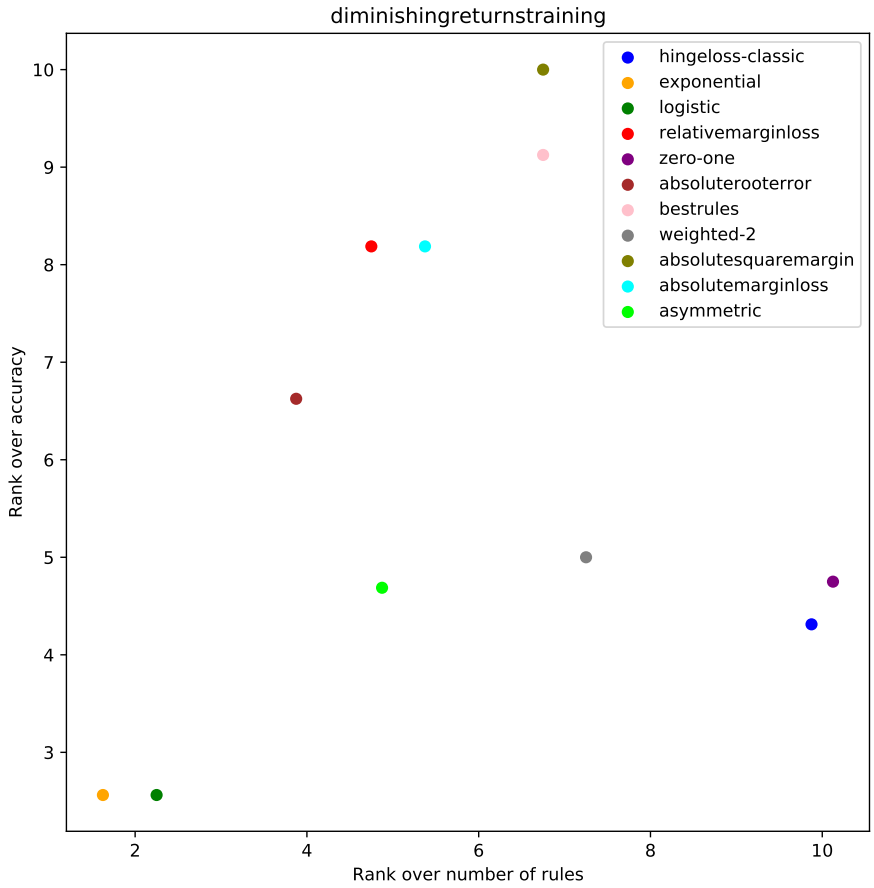


Figure 7.9: Scatter diagram to show the relation between number of rules and accuracy at the stopping criterion. The ranks are averaged on both axis.

the point of diminishing returns is reached on trainingdata and the improvements with every rule slow down while there is still a lot of accuracy to be gained on testdata.

Figure 7.8 and figure 7.9 show that this is the case for almost all methods and baseline

methods. It is not the case for weighted covering which ranks better in both number of rules and achieved accuracy on the testdata ranking. It seems like a strategy based around the coverage of instances is less susceptible to overfitting. Other loss functions that improve in their ranking on testdata compared to their ranking on training data are asymmetric and absoluterooterror which are anew consistently better than absolutesquaremargin. The loss functions which achieve the overall best results with the diminishing returns are logistic loss and exponential loss. They suffer from worse average ranks on testdata though.

7.3.4 Breakeven with Random Rules

The comparison to random rules is compelling because it gives insight in the effectiveness of a strategic methodology at all. However, the results vary a lot between datasets and loss functions. Figure 7.10 gives an example on how random rules can catch up with asymmetric as the loss function. The break even point states that it is possible to extract up to 173 rules to have a good accuracy with few rules.

The loss function in figure 7.10 has a phase later in the rule selection when it performs noticeably better than random rules. The advantage of the breakeven point is that differences like this later in the rule selection process have no effect on the measurement and comparison.

Figure 7.11 shows an example where the loss function outperforms random rules throughout all simplified random forest sizes. The breakeven stopping criterion cannot distinguish loss functions like this since they always achieve the first rank.

Furthermore, there is the possibility that a loss function chooses worse than average rules as the first few rules (e.g. in figure 7.12). In this case the method or loss function always performs as the worst method.

The simpler loss functions zero-one and hingeloss-classic perform the worst with this stopping criterion. The table in figure 7.13 shows that they achieve consistently one of the worst breakeven points.

The best function by this measurement is harder to identify. Exponential loss achieves overall good results but is one of the worst loss functions on the hepatitis dataset. The overall best loss function for a preferably late break even point is absoluterooterror.

Figure 7.10: Example comparison between gain covering and random rules with asymmetric on heart-statlog.arff. The dotted lines represent the best and worst accuracies over all random runs and over cross-validation folds.

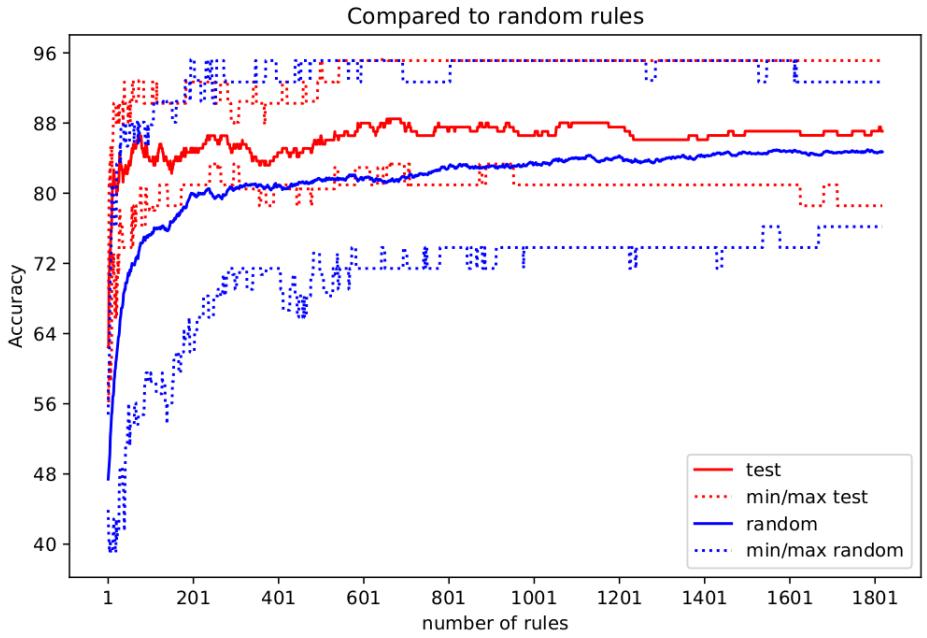


Figure 7.11: Example comparison between gain covering and random rules with exponential on sonar.arff. The dotted lines represent the best and worst accuracies over all random runs and over cross-validation folds.

Figure 7.12: Example comparison between gain covering and random rules with best rules on heart-statlog.arff. The dotted lines represent the best and worst accuracies over all random runs and over cross-validation folds.

Breakeven

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar	
absolutemarginloss	8.5	1.5	4.0	9.0	8.5	6.0	1.5	8.0	47.0
absoluterooterror	1.0	1.5	4.0	5.0	1.5	3.5	3.0	5.0	24.5
absolutesquaremargin	8.5	7.5	4.0	6.5	8.5	9.0	7.0	9.0	60.0
asymmetric	2.0	9.0	4.0	4.0	1.5	7.0	1.5	4.0	33.0
exponential	5.0	5.0	4.0	1.0	4.0	1.0	9.0	1.5	30.5
hingeloss-classic	8.5	10.5	9.5	10.5	8.5	11.0	10.5	10.5	79.5
logistic	4.0	4.0	4.0	2.0	3.0	3.5	8.0	3.0	31.5
relativemarginloss	8.5	3.0	4.0	6.5	8.5	2.0	6.0	7.0	45.5
zero-one	8.5	10.5	9.5	10.5	8.5	10.0	10.5	10.5	78.5
bestrules	8.5	7.5	9.5	8.0	8.5	8.0	5.0	6.0	61.0
weighted-2	3.0	6.0	9.5	3.0	5.0	5.0	4.0	1.5	37.0

Figure 7.13: Ranked scores over the number of rules necessary to surpass the random forest with summed up ranks (last column).

7.3.5 Surpass Random Forest

Firstly, it is necessary to find out whether it is possible to surpass the accuracy by the whole random forest at all. The table below shows the rules necessary to surpass the whole random forest. Despite this being more difficult in some datasets (e.g. the titanic and ionsphere datasets) it is achieved most of the time.

Because the baseline methods surpass the random forest as well, it can be examined which methods and loss functions surpass the random forest with fewer rules. Table 7.14 shows the results ranked.

Surpass

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar	
absolutemarginloss	6	9	3	7	9	6	7	7	54.0
absoluterooterror	5	3	6	7	5	7	1	5	39.0
absolutesquaremargin	11	11	5	7	10	9	11	9	73.0
asymmetric	4	6	6	2	4	8	2	4	36.0
exponential	2	1	6	3	3	1	5	1	22.0
hingeloss-classic	10	3	6	6	7	5	10	10	57.0
logistic	2	1	6	3	2	10	9	1	34.0
relativemarginloss	7	8	3	7	6	2	6	6	45.0
zero-one	9	3	6	5	7	4	8	10	52.0
bestrules	8	10	2	7	11	11	3	8	60.0
weighted-2	1	7	1	1	1	3	4	3	21.0

Figure 7.14: Ranked scores over the number of rules necessary to surpass the random forest with summed up ranks (last column).

As in other stopping criteria and in the measurements by area under curve, the simple loss functions zero-one and hingeloss-classic achieve relatively bad results. Furthermore absolutesquaremargin has the overall worst results while the loss functions with the opposite strategy (asymmetric and absoluterooterror) perform significantly better.

The baseline methods perform very dissimilar to each other. While weighted covering achieves the overall best results in surpassing the random forest the best rules baseline method is within the worst methods to surpass the random forest.

	heart-statlog	breast-cancer	titanic	ionosphere	vote	labor	hepatitis	sonar
absolutemarginloss	137.0	107.0	18.0	-1.0	1763.0	533.0	390.0	1518.0
absoluterooterror	38.0	13.0	-1.0	-1.0	918.0	557.0	11.0	1354.0
absolutesquaremargin	1480.0	3301.0	19.0	-1.0	2616.0	624.0	1281.0	1776.0
asymmetric	27.0	22.0	-1.0	13.0	670.0	560.0	49.0	1264.0
exponential	25.0	10.0	-1.0	94.0	458.0	33.0	266.0	60.0
hingeloss-classic	802.0	13.0	-1.0	1352.0	1376.0	347.0	784.0	-1.0
logistic	25.0	10.0	-1.0	94.0	457.0	1011.0	535.0	60.0
relativemarginloss	140.0	104.0	18.0	-1.0	1331.0	118.0	307.0	1514.0
zero-one	728.0	13.0	-1.0	1340.0	1376.0	189.0	423.0	-1.0
bestrules	214.0	3194.0	15.0	-1.0	-1.0	-1.0	51.0	1749.0
weighted-2	9.0	23.0	13.0	6.0	103.0	137.0	109.0	265.0

Table 7.4: The number of rules necessary to surpass the random forest. A number of rules of -1 means that the loss function did not receive an accuracy higher than the underlying random forest.

7.4 Evaluation

The different measurements above have shown that there is no unambiguously best loss function. Furthermore the differences between loss functions are quite often so narrow that trends are only noticeable when building up ranks.

One consistent pairwise comparison is between loss functions that have increasingly growing rewards and penalties (absolutesquaremargin) and loss functions that have stagnantly growing rewards and penalties(asymmetric and absoluterooterror). Asymmetric and absoluterooterror can beat absolutesquaremargin pretty reliably.

The reason for this is unclear. One hypothesis is that the loss function is overly focused on correcting previous mistakes. The indication for this is a look into the coverage of instances which often has notable humps later in the rule selection. In the appendix are individual evaluations of dataset and loss function combinations. The last page of each contains graphs that show what percentage is covered by every newly added rule and how many instances the new rule tries to correct (an instance that got an incorrect classification from the already selected rules but a correct classification by the newly selected rule in every step).

Absoluterooterror and asymmetric stand out as relatively good performing loss functions as well. Especially in the measurements by area under curve they are leading regardless of the used weight function. Other prominent loss functions are logistic loss and exponential

loss. They stand out as good in surpassing the random forest and ideal at the point of diminishing returns on testdata and trainingdata.

In contrast to that, the simpler loss functions such as zero-one and hingeloss-classic are trailing in many measurements.

The remaining loss functions absolutemarginloss and relativemarginloss are overall unremarkable. As absolutemarginloss does not stand out as clearly under performing or outperforming absoluterooterror and absolutesquaremargin it does not contradict the conclusion that the stagnantly growing rewards and penalties asymmetric and absoluterooterror are advantageous.

The baseline method best rules is trailing in many measurements comparable to zero-one and hingeloss-classic. the baseline method weighted covering, on the other hand, can keep up with gain covering and even leads in some measurements such as the number of rules for the full coverage stopping criterion or when surpassing the random forest.

8 Conclusion

The results from the measurements by area under curve and several stopping criteria show that it is possible to achieve better classification than existing baseline approaches. Furthermore the presented method can beat the whole random forest reliably.

However, no loss function turned out to be consistently the best loss function and no loss function is able to beat baseline approaches unambiguously every time. Especially weighted covering can often keep up with the method presented in this work.

In conclusion, differences between loss functions and comparisons to baseline approaches are not sufficiently consistent to finish the search for an ideal trade-off between interpretability and accuracy yet. The improvement over baseline methods is not significant enough or worth the effort of implementation for a serious real world application.

8.1 Future Work

Simplifying random forests is still a field with many possibilities. Optimizing for the best possible classification with few rules and surpassing the random forest to find the best accuracy possible show great potential. There are still many parameters that can be tweaked and different ways a random forest can be processed into derivative AI models.

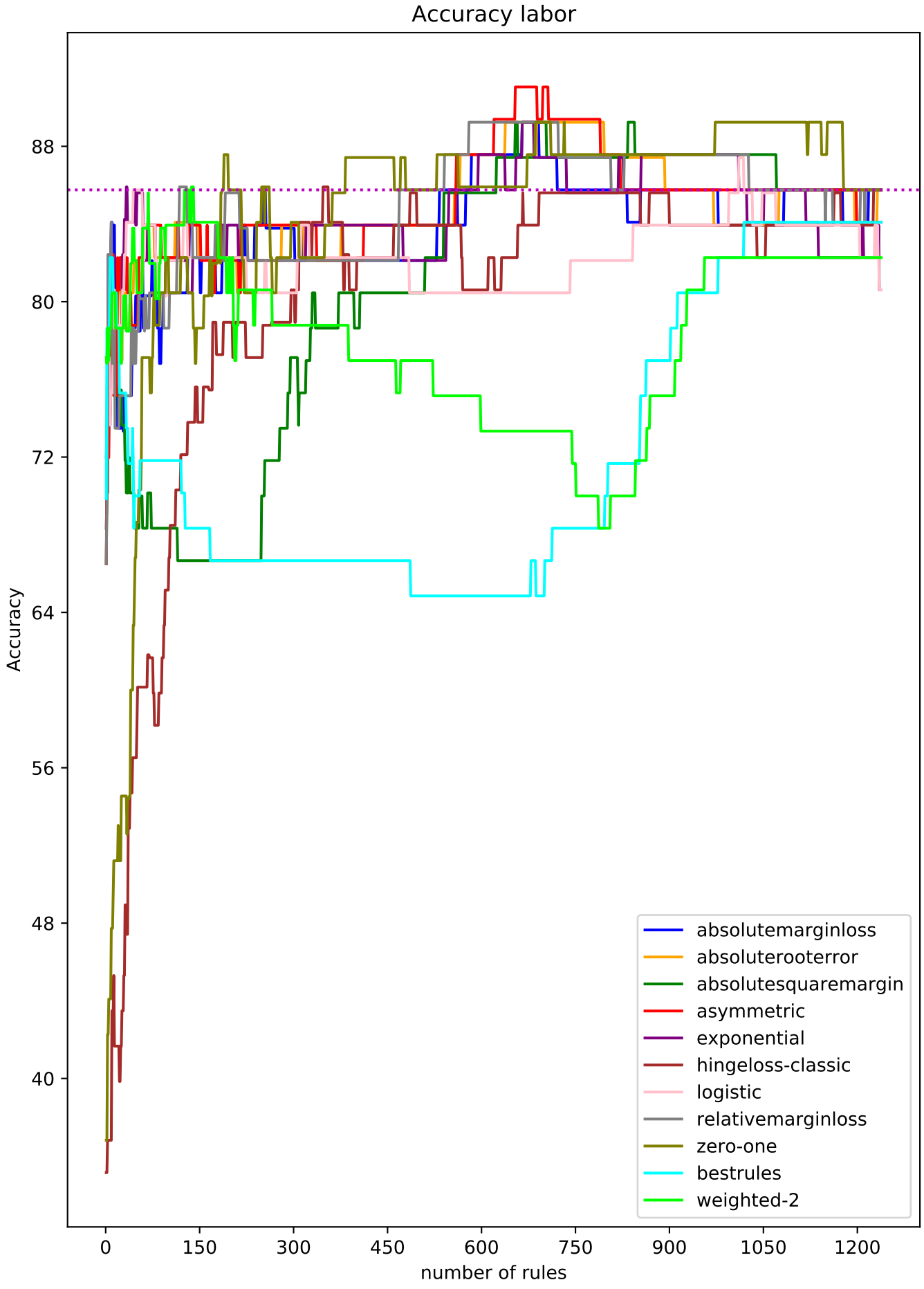
One possible approach could be possible to measure the similarity between rules in order to optimize for diversity or avoid contradicting rules that would more likely result in nonsense-interpretations. Furthermore, the rules could be tweaked themselves by modifying conditions, adding or removing conditions as well as combining rules. Another idea would be to build up new decision trees with a subset of rules or tweaked rules.

But most importantly, before a new technology from this field takes off in the real world it is unavoidable to quantify how good the interpretability of simplified random forests really are - preferably with expert surveys in their respective fields.

9 Appendix

The following graphs show the progess in accuracy of a rule selection for the labor dataset. The dotted line represents the accuracy by the whole random forest.

The curves do not end up at the same accuracy because the graphics only include simplified random forests accuracies that appear in all folds of the cross-validation. This is due to the nature of the underlying random forests which result in different decision trees with different depth when training with various training data. Regardless of these technical hurdles, the course of accuracy is backed up by a 5-fold cross-validation and the different loss functions per dataset can be compared.


Accuracy labor
Accuracy
88
80
72
64
56
48
40
0
150
300
450
600
750
900
1050
1200
number of rules
absolutemarginloss
absoluterooterror
absolutesquaremargin
asymmetric
exponential
hingeloss-classic
logistic
relativemarginloss
zero-one
bestrules
weighted-2

The following contains example analysis over individual dataset and loss function combinations.

sonar.arff-asymmetric.csv

Number of rules: 1818.0

Number of traininginstances: 832.0 Number of testinstances: 208.0

Accuracy of the random forest: 85.1

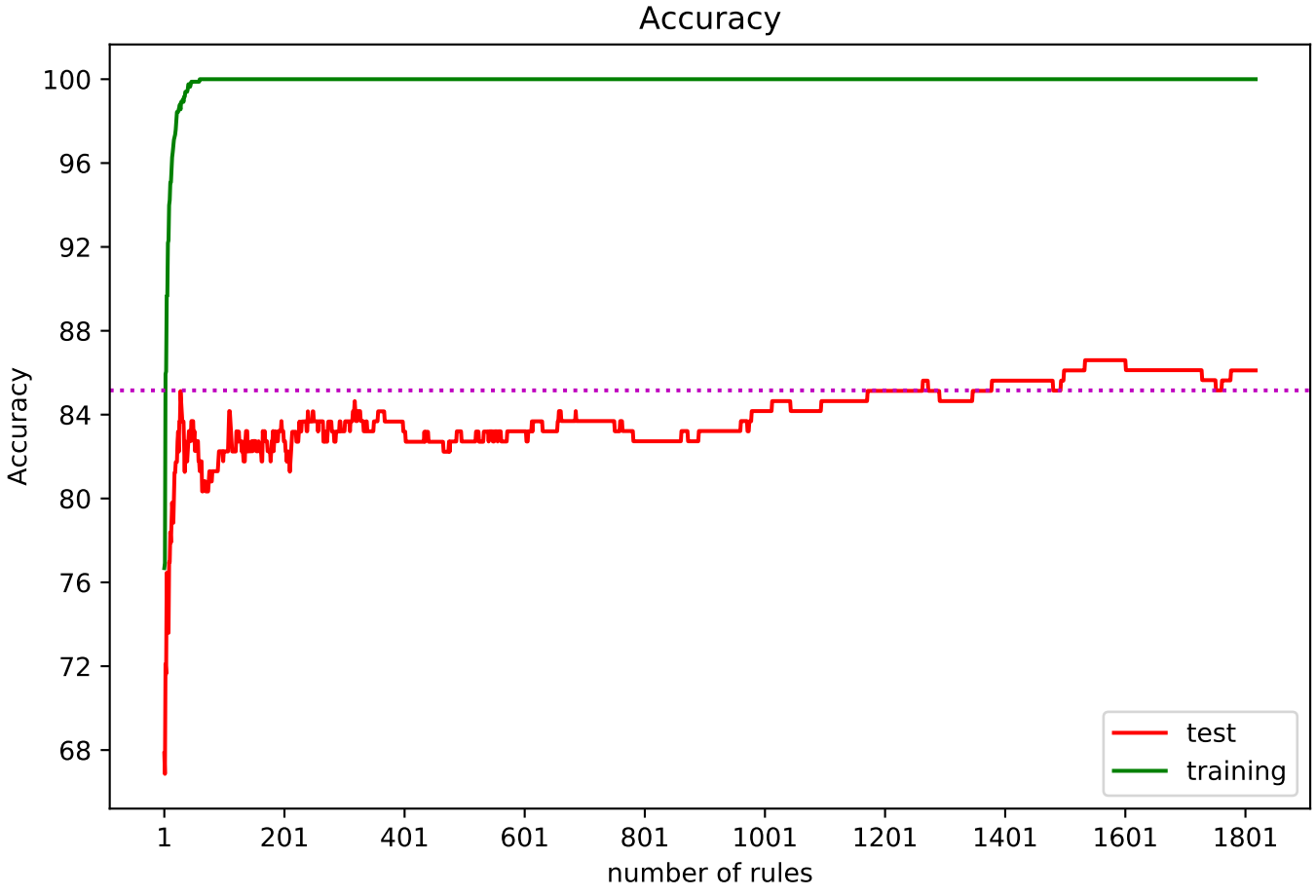
Score integral: 0.80 Score linear: 0.83 Score dcg: 0.80

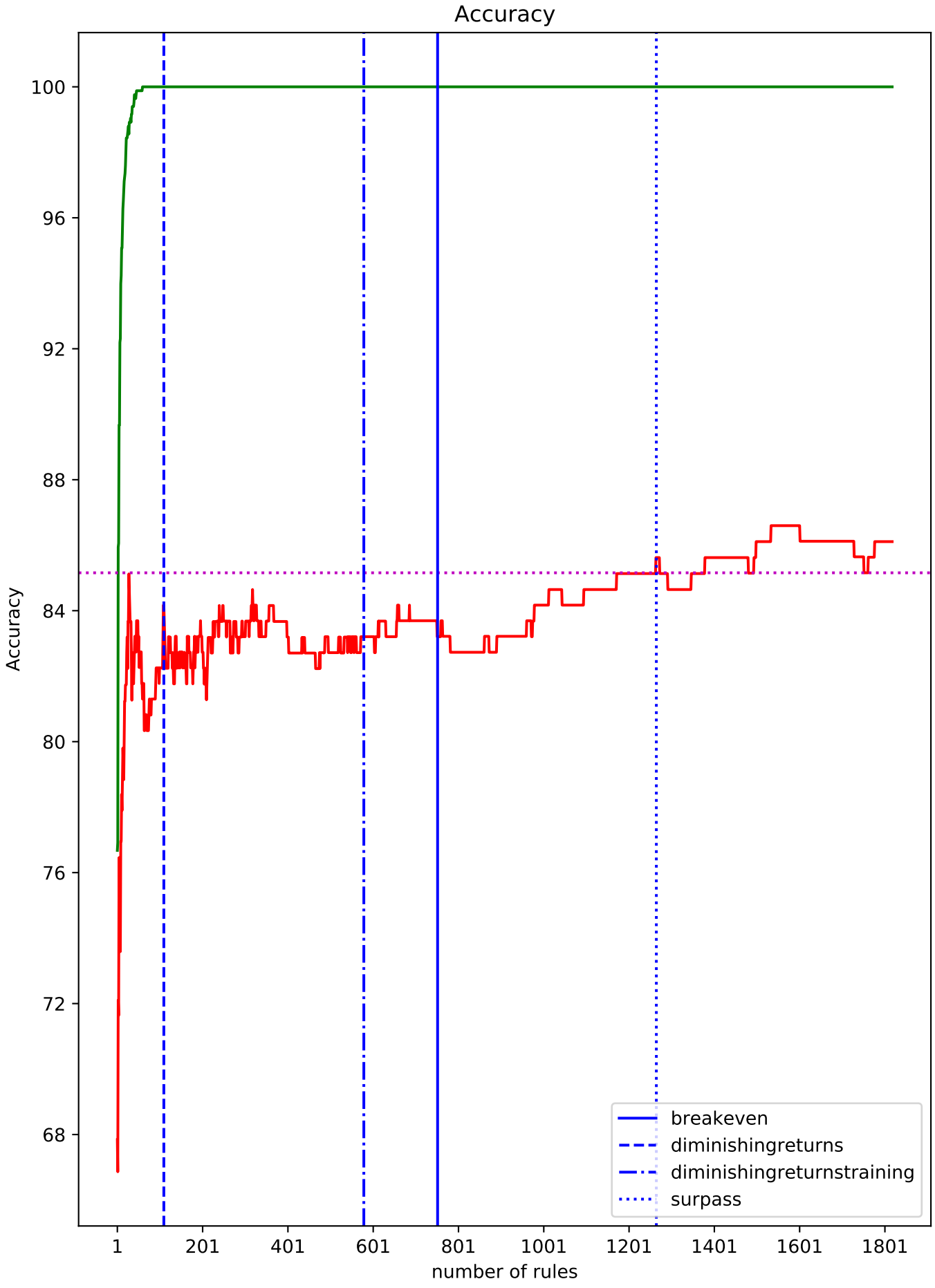
Breakeven: 751.0 rules (Accuracy: 83.2)

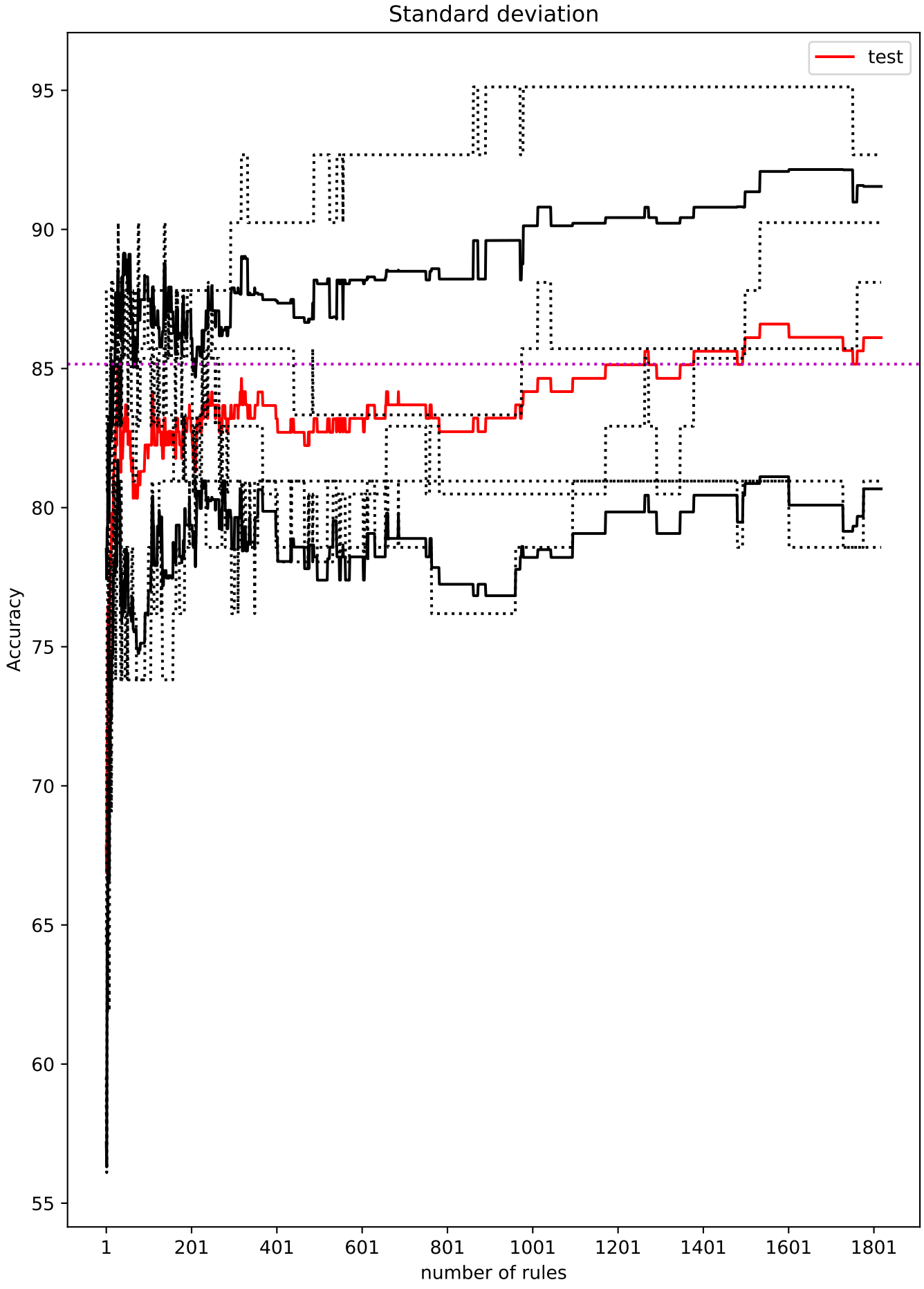
Diminishing returns: 109.0 rules (Accuracy: 84.1)

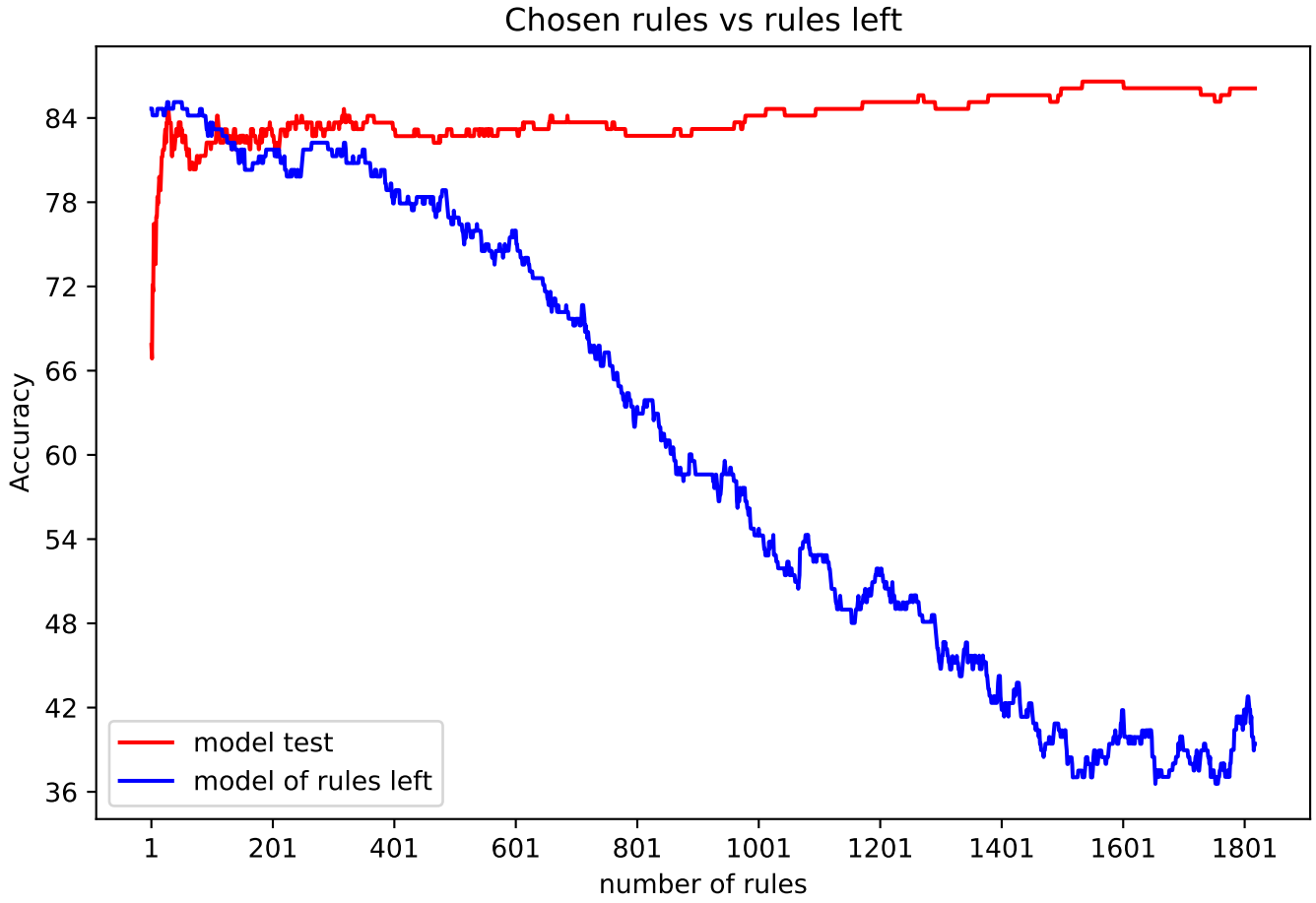
Diminishing returns on trainingdata: 578.0 rules (Accuracy: 83.2)

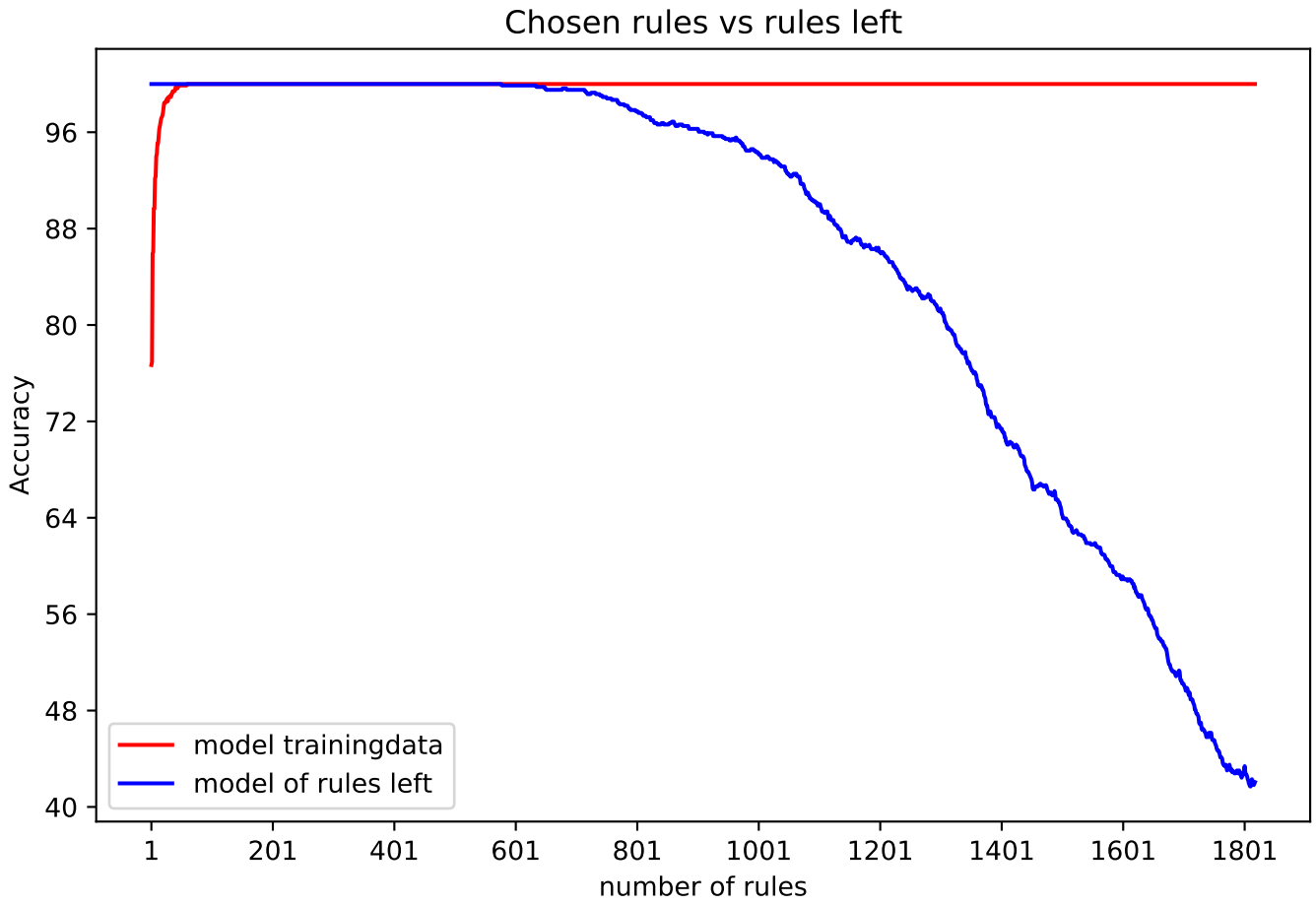
Surpass random forest: 1264.0 rules

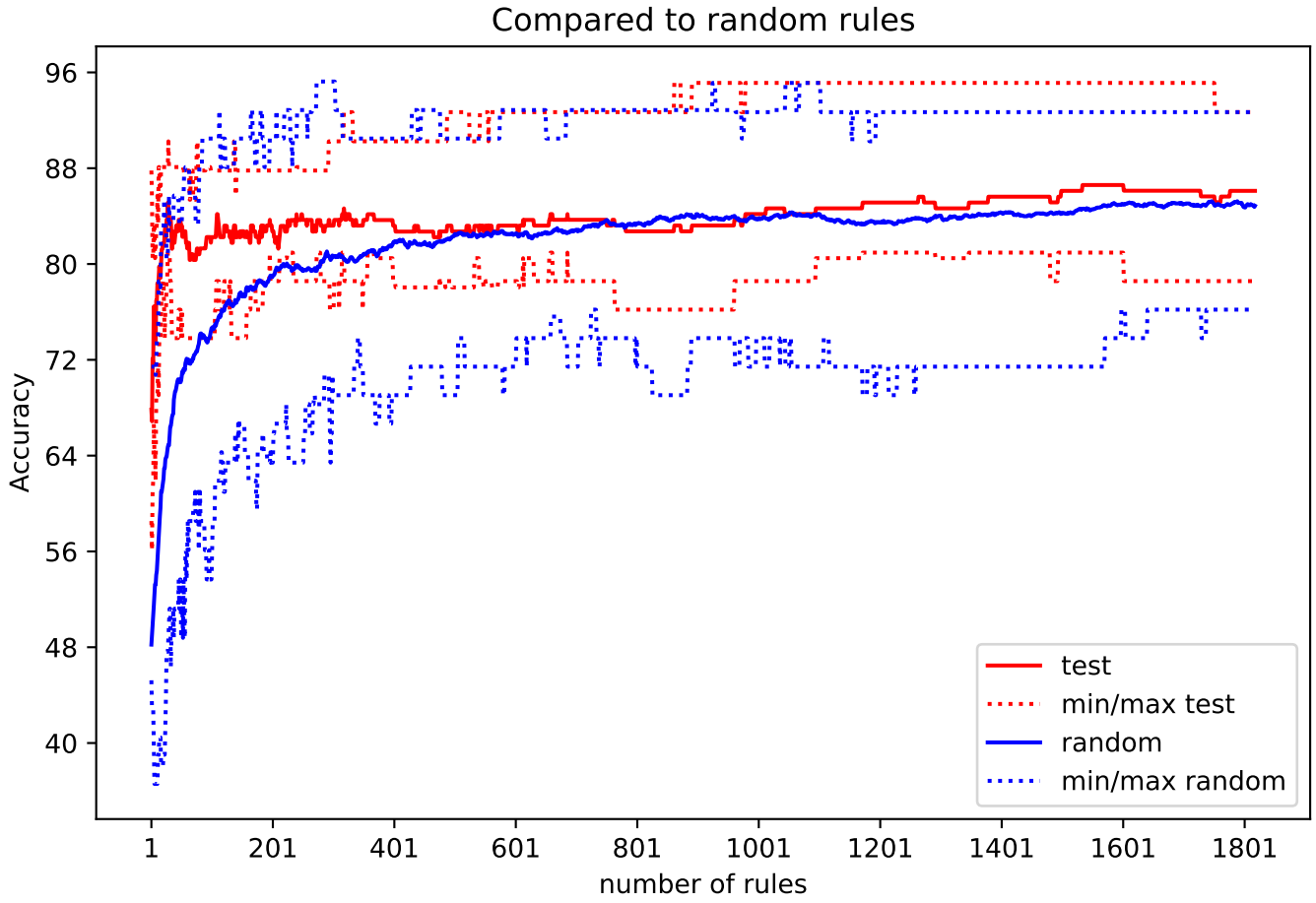


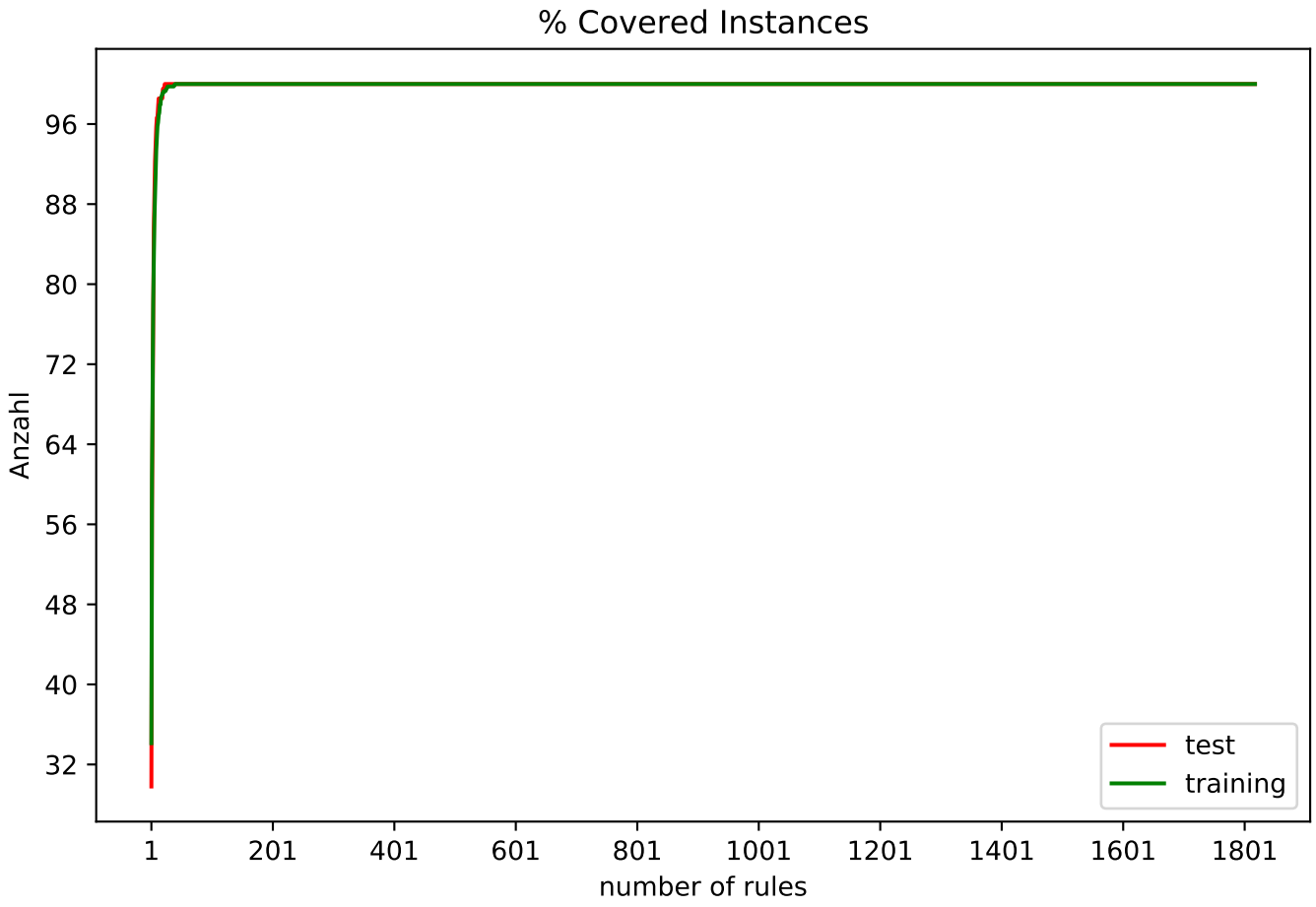

Accuracy
100
96
92
88
84
80
76
72
68
Accuracy
1
201
401
601
801
1001
1201
1401
1601
1801
number of rules
breakeven
diminishingreturns
diminishingreturnstraining
surpass

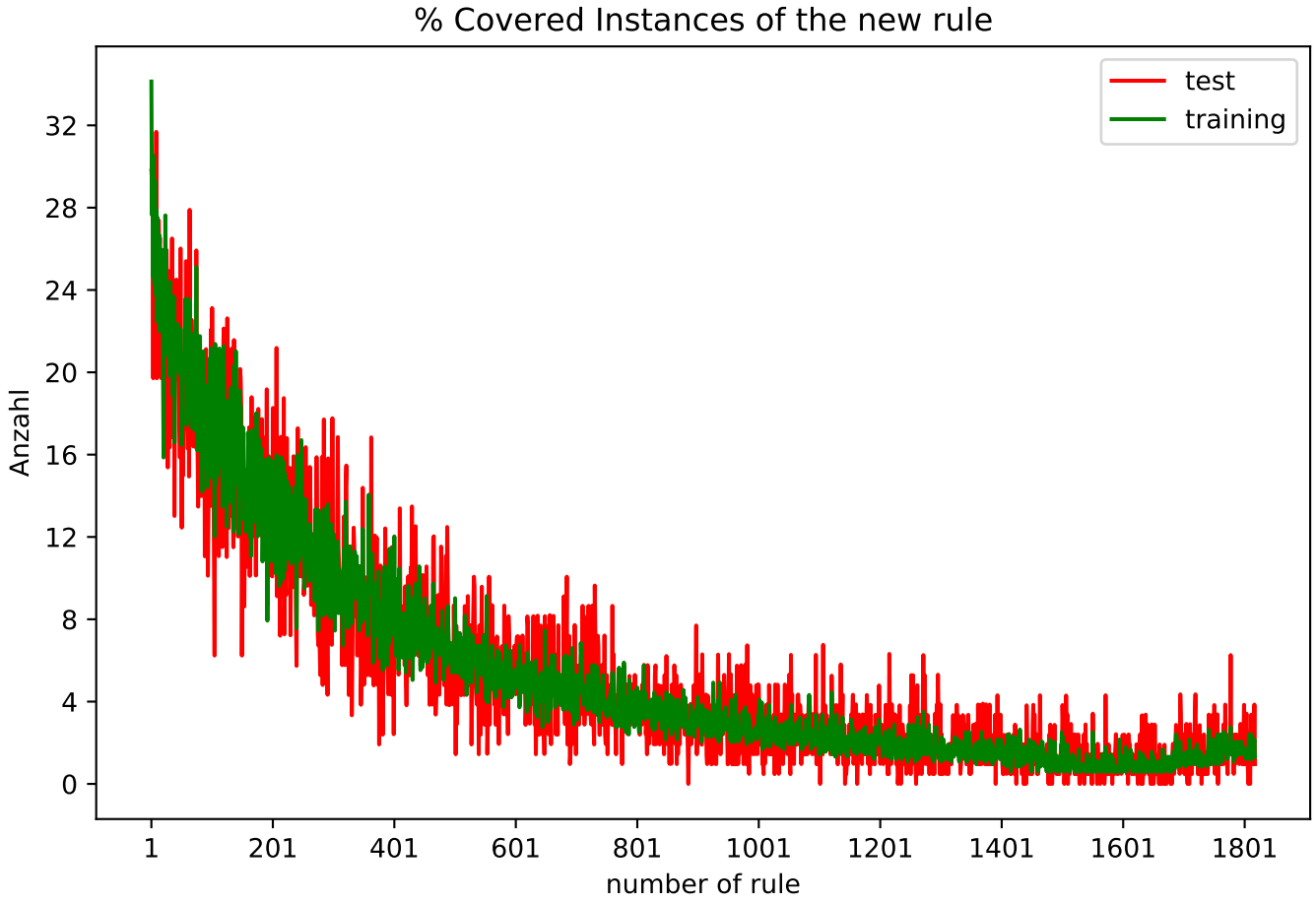

Standard deviation
test
95
90
85
80
75
70
65
60
55
Accuracy
1
201
401
601
801
1001
1201
1401
1601
1801
number of rules

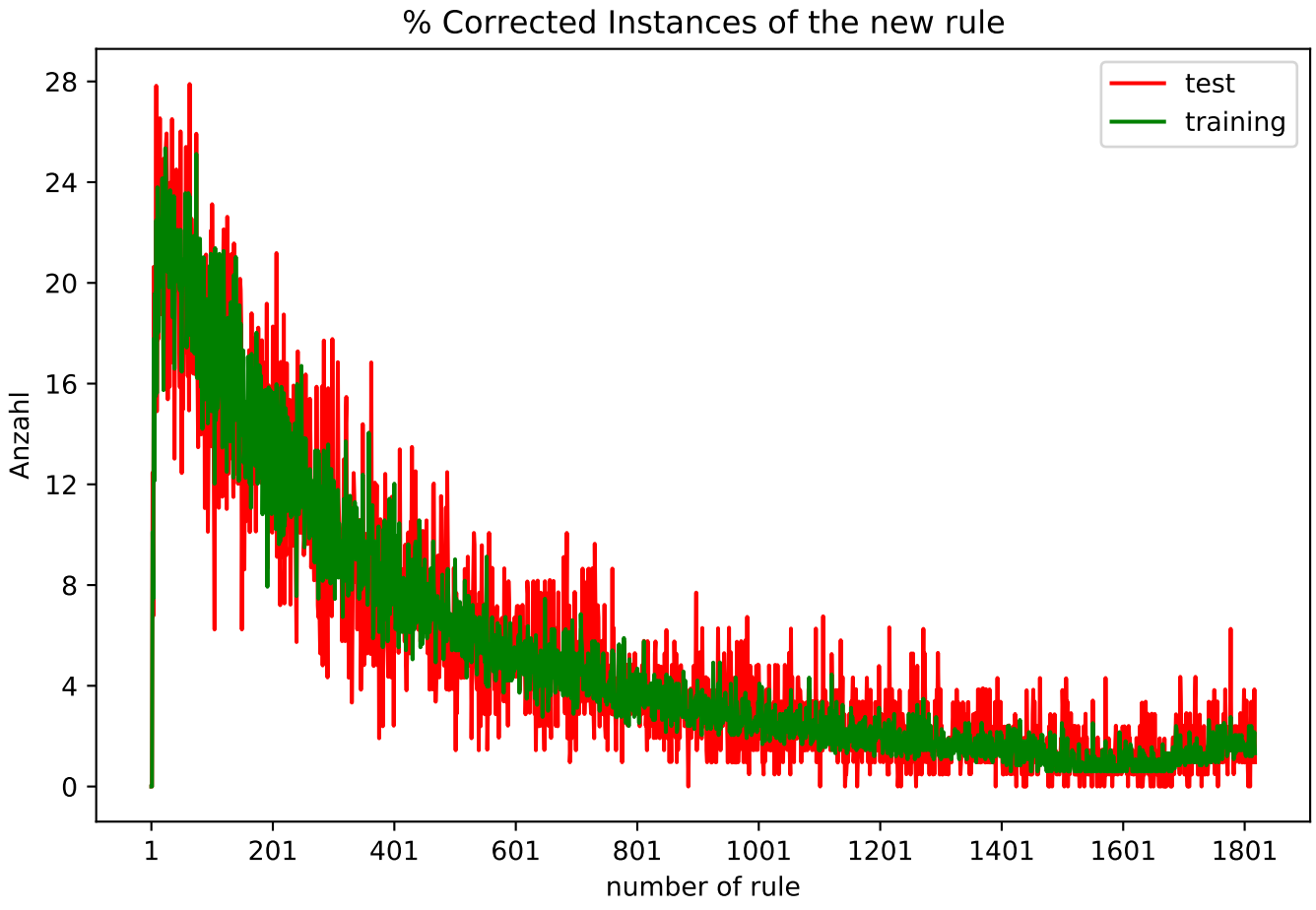

Chosen rules vs rules left
Accuracy
number of rules
model test
model of rules left
84
78
72
66
60
54
48
42
36
1
201
401
601
801
1001
1201
1401
1601
1801


Chosen rules vs rules left
Accuracy
number of rules
model trainingdata
model of rules left
96
88
80
72
64
56
48
40
1
201
401
601
801
1001
1201
1401
1601
1801


Compared to random rules
Accuracy
number of rules
test
min/max test
random
min/max random


% Covered Instances
Anzahl
number of rules
test
training


% Covered Instances of the new rule
test
training
Anzahl
0
4
8
12
16
20
24
28
32
1
201
401
601
801
1001
1201
1401
1601
1801
number of rule


% Corrected Instances of the new rule
test
training
Anzahl
0
4
8
12
16
20
24
28
1
201
401
601
801
1001
1201
1401
1601
1801
number of rule

sonar.arff-absolutesquaremargin.csv

Number of rules: 1818.0

Number of traininginstances: 832.0 Number of testinstances: 208.0

Accuracy of the random forest: 85.1

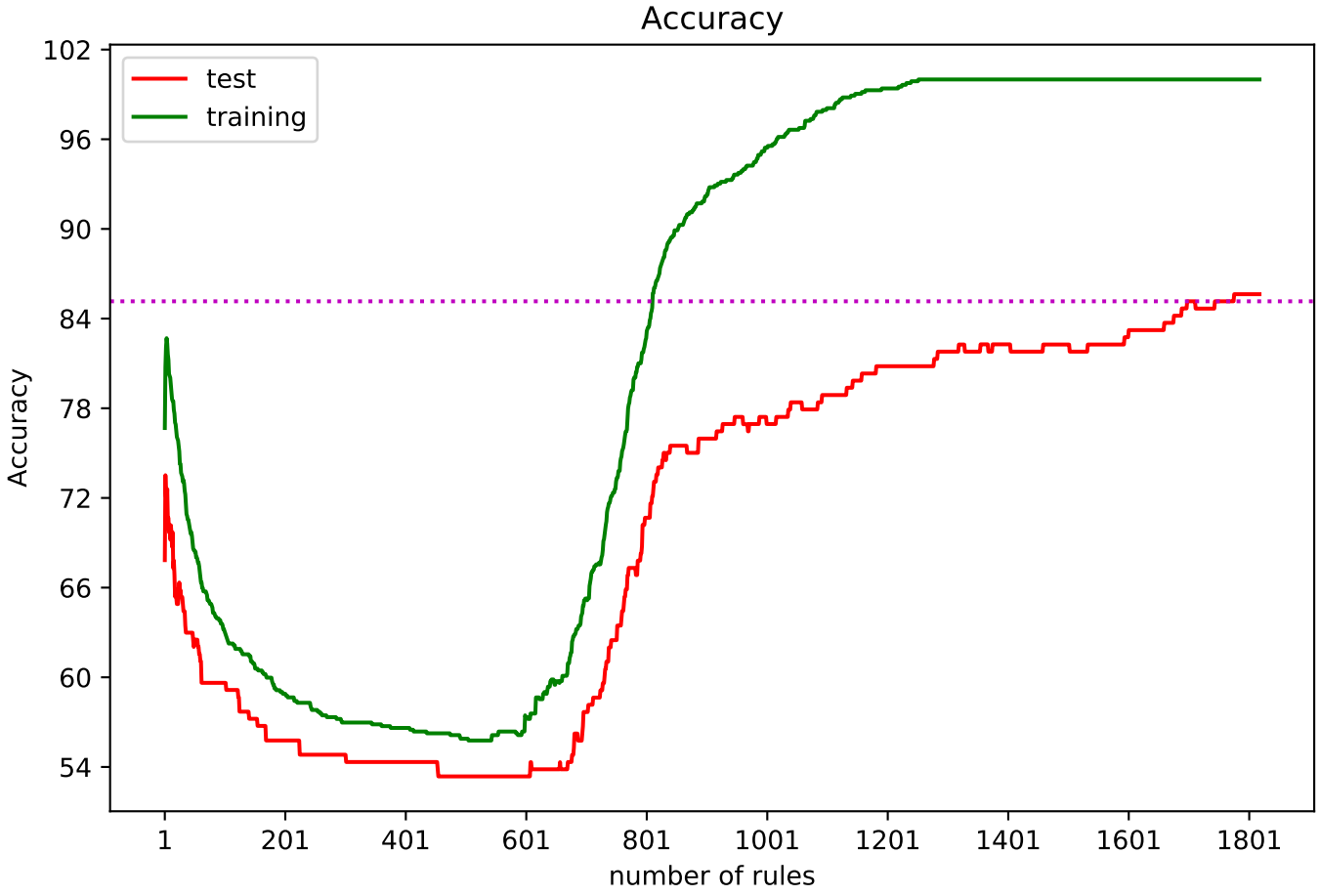
Score integral: 0.67 Score linear: 0.64 Score dcg: 0.66

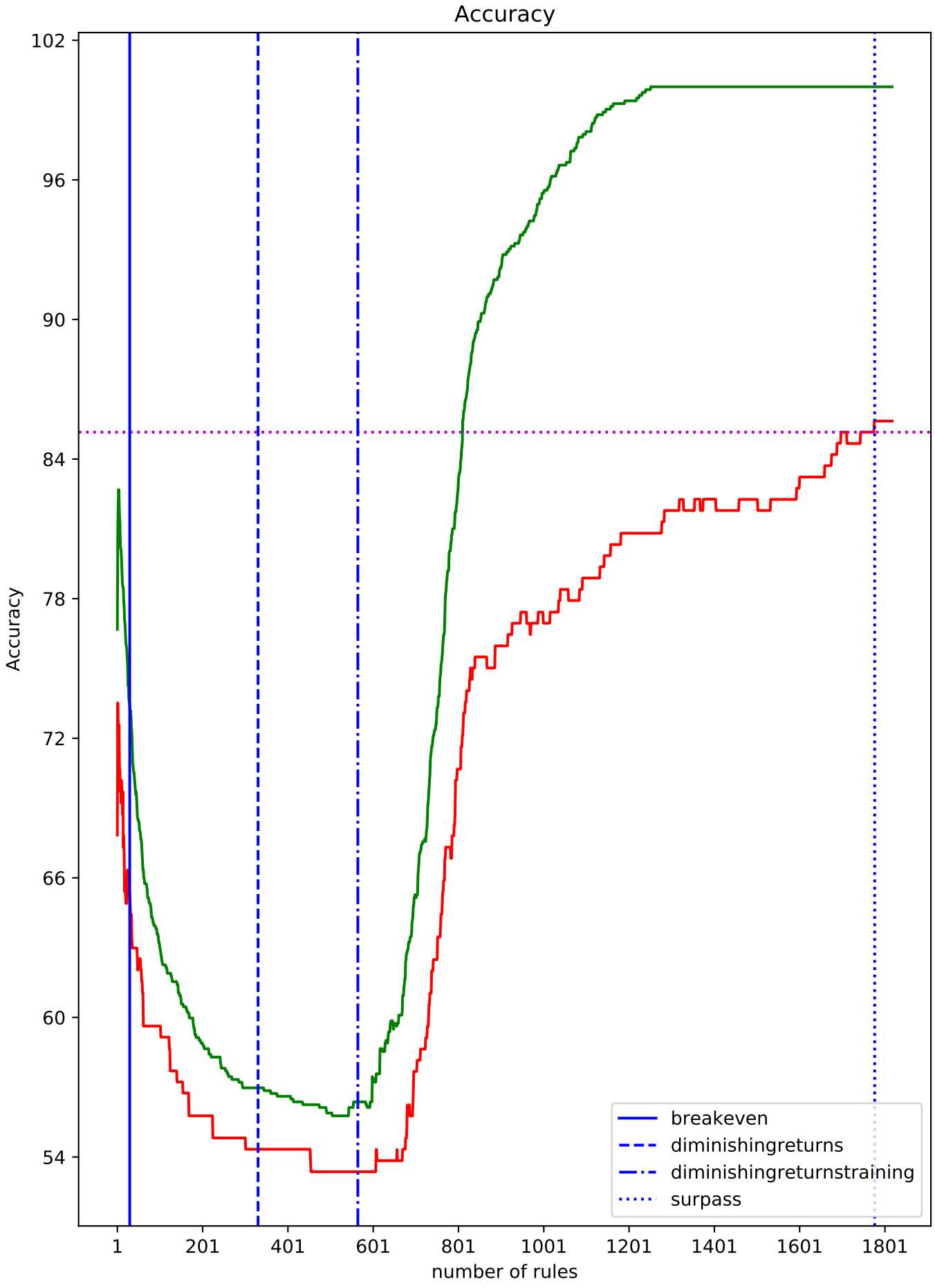
Breakeven: 29.0 rules (Accuracy: 65.3)

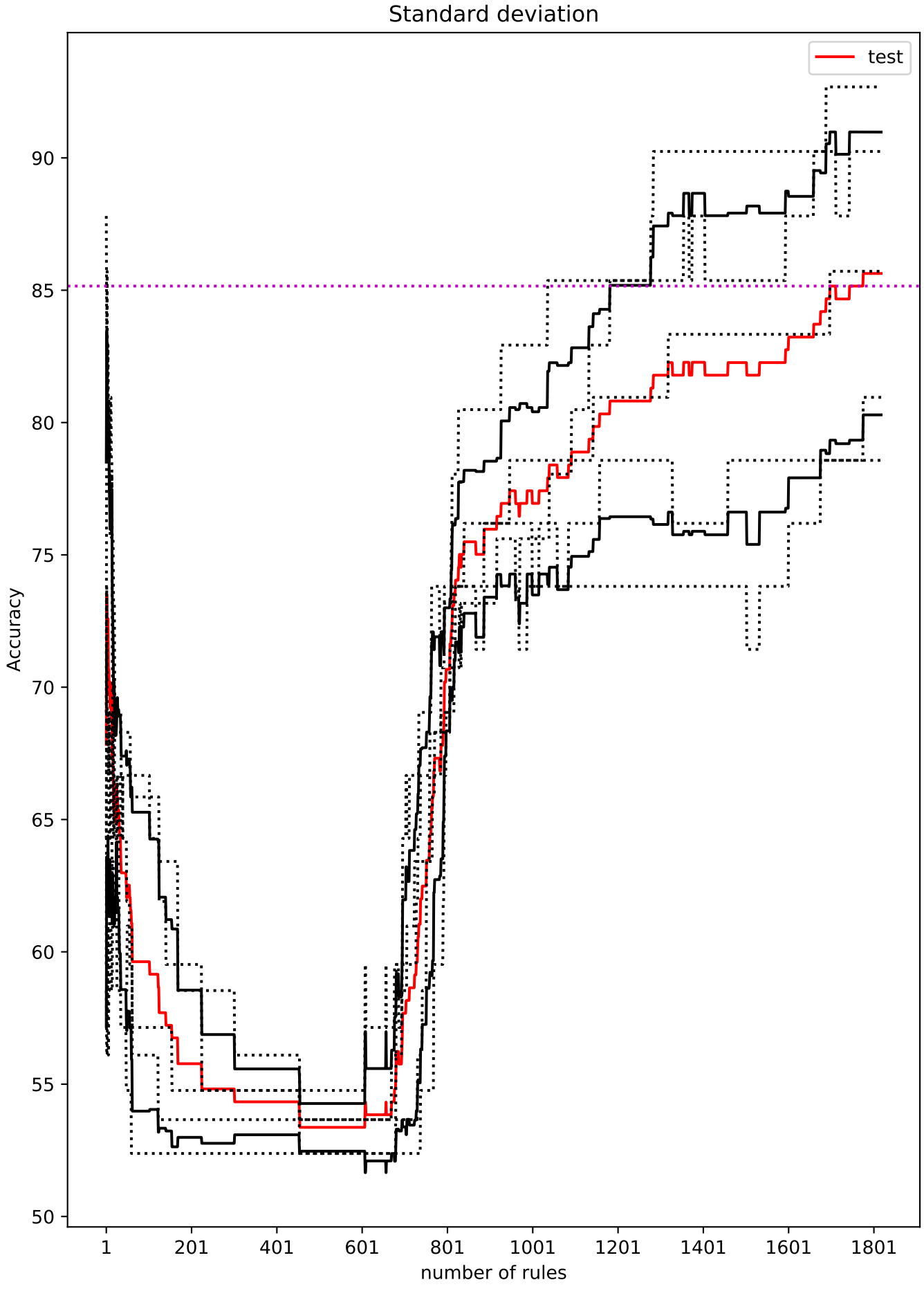
Diminishing returns: 330.0 rules (Accuracy: 54.3)

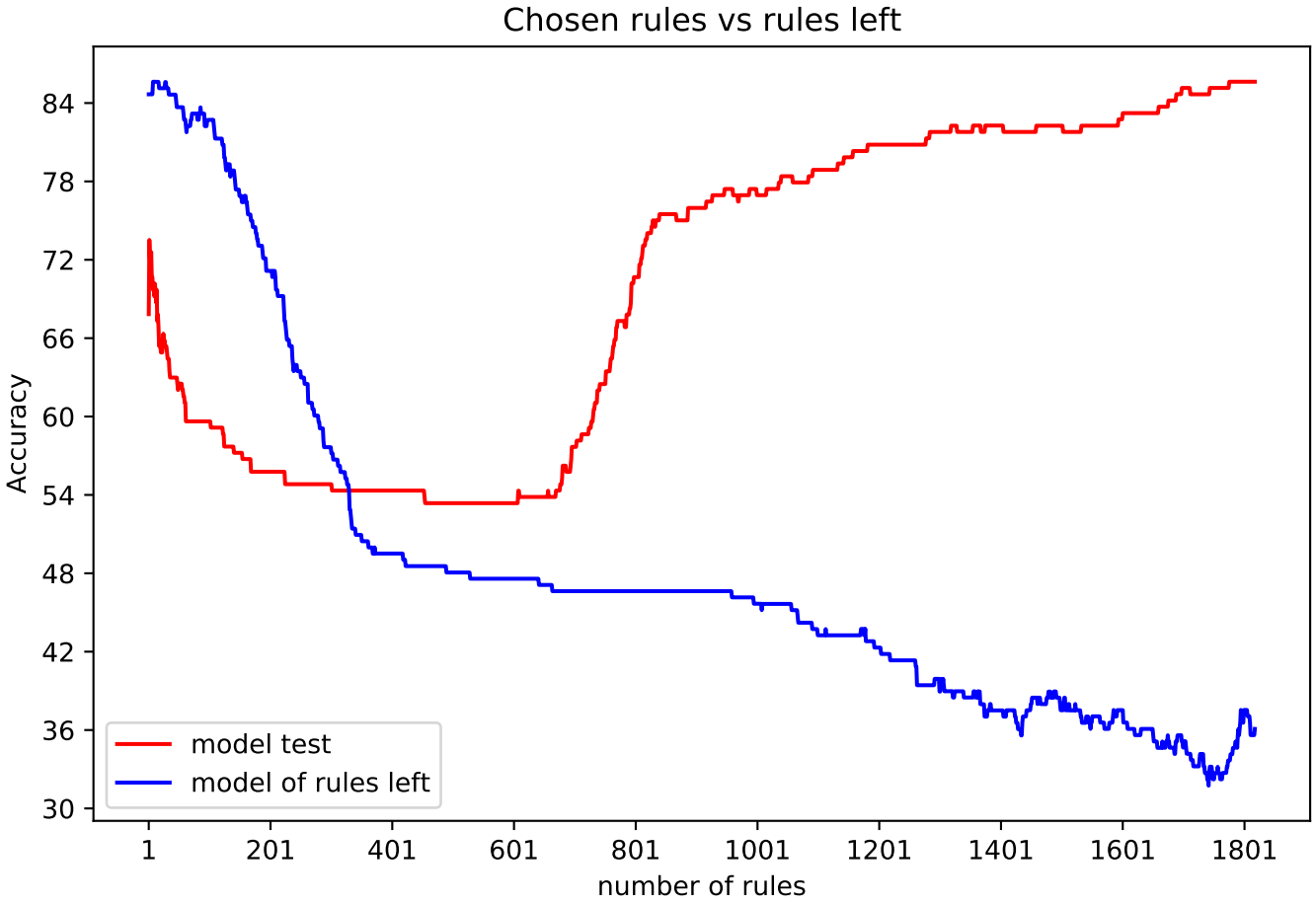
Diminishing returns on trainingdata: 564.0 rules (Accuracy: 56.3)

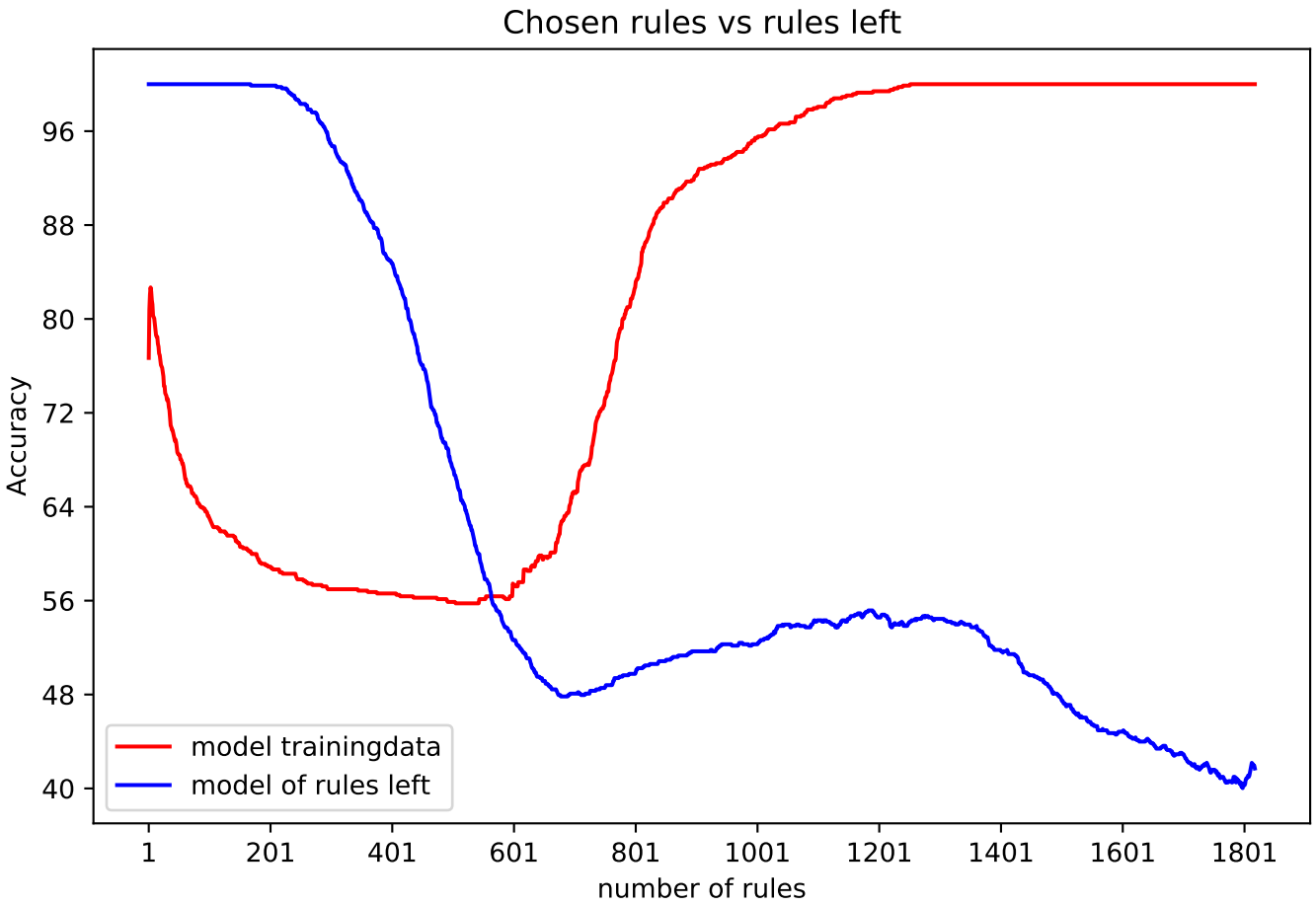
Surpass random forest: 1776.0 rules

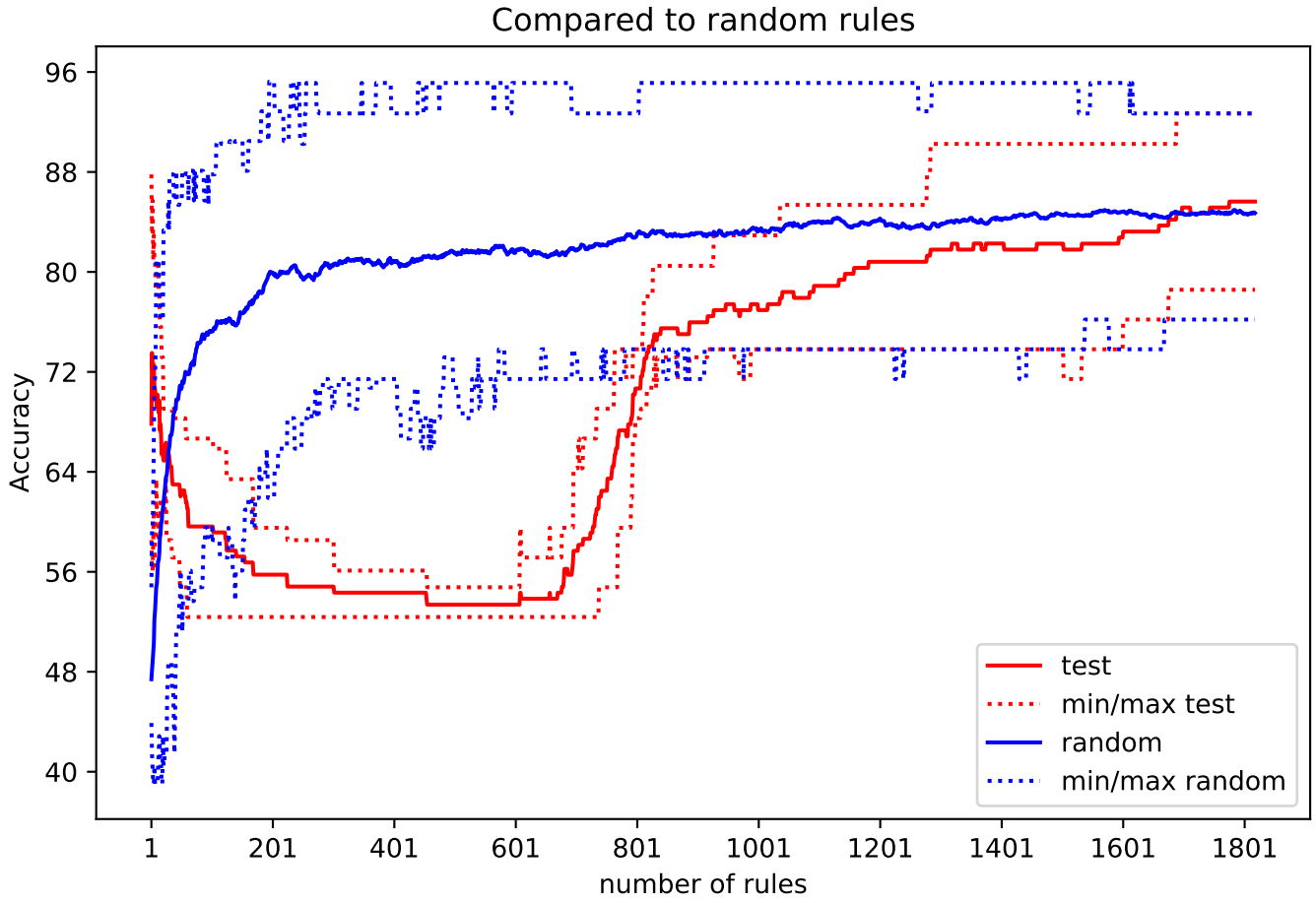


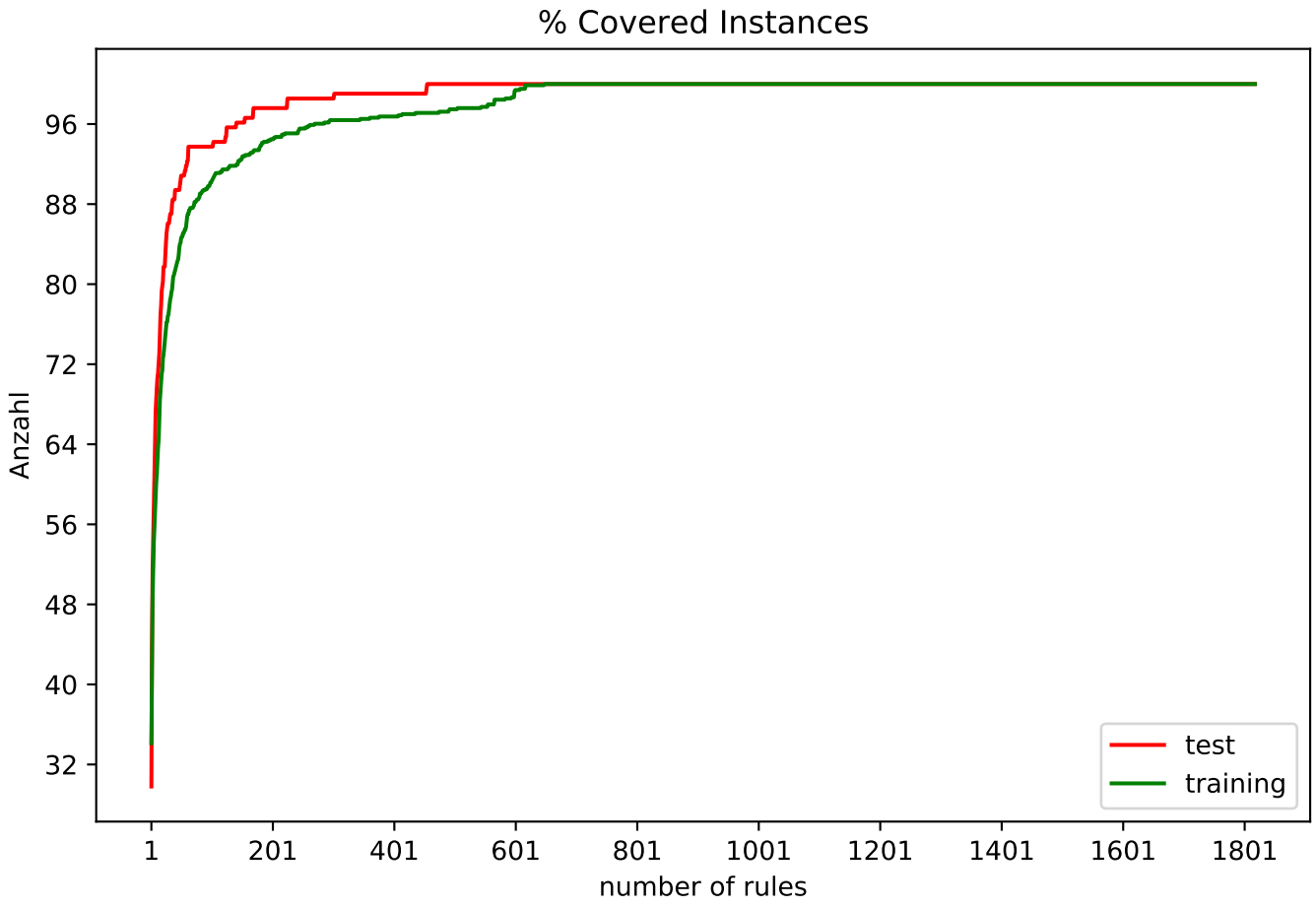

Accuracy
102
96
90
84
78
72
66
60
54
Accuracy
1
201
401
601
801
1001
1201
1401
1601
1801
number of rules
breakeven
diminishingreturns
diminishingreturnstraining
surpass

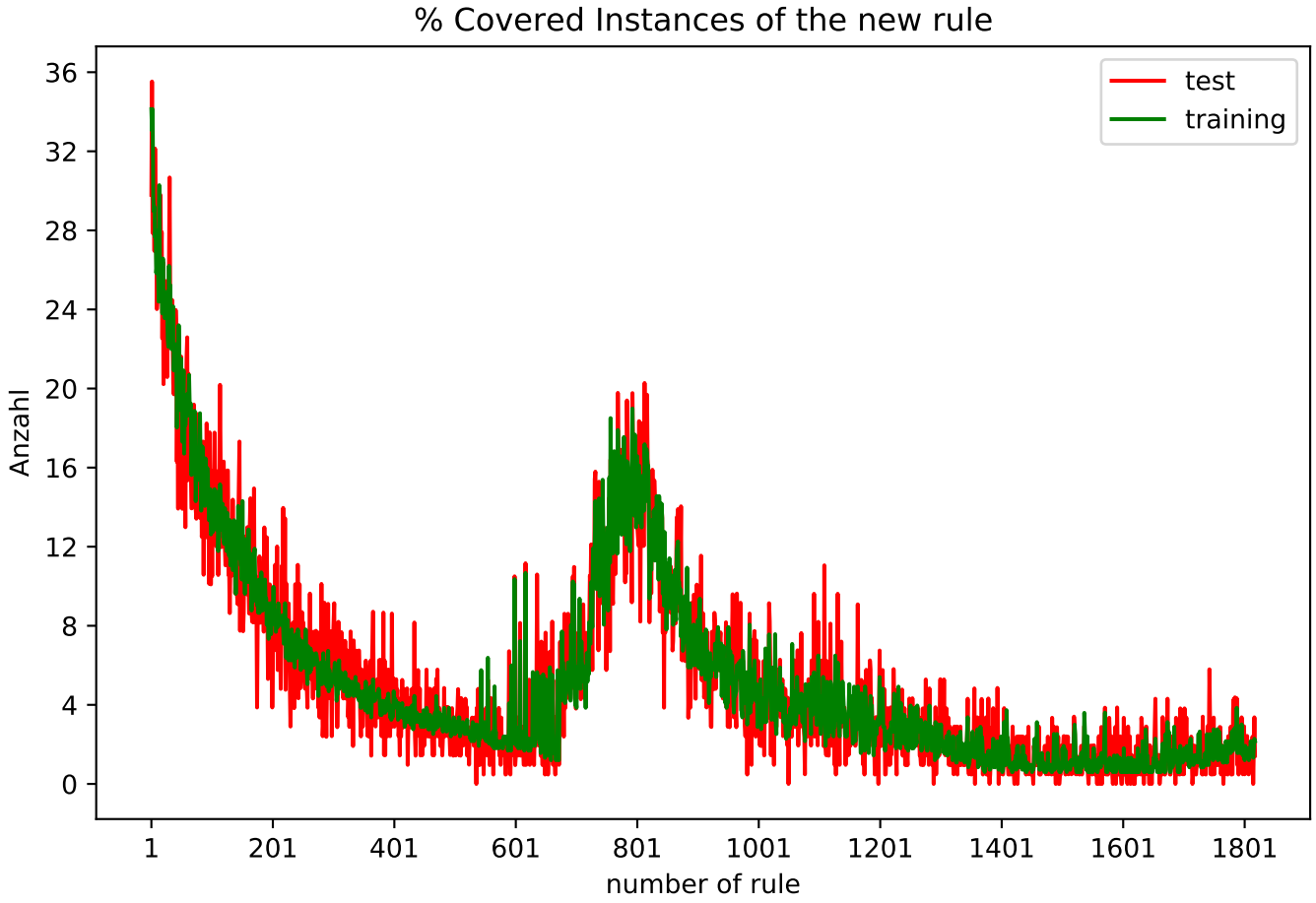

Standard deviation
test
Accuracy
number of rules
90
85
80
75
70
65
60
55
50
1
201
401
601
801
1001
1201
1401
1601
1801

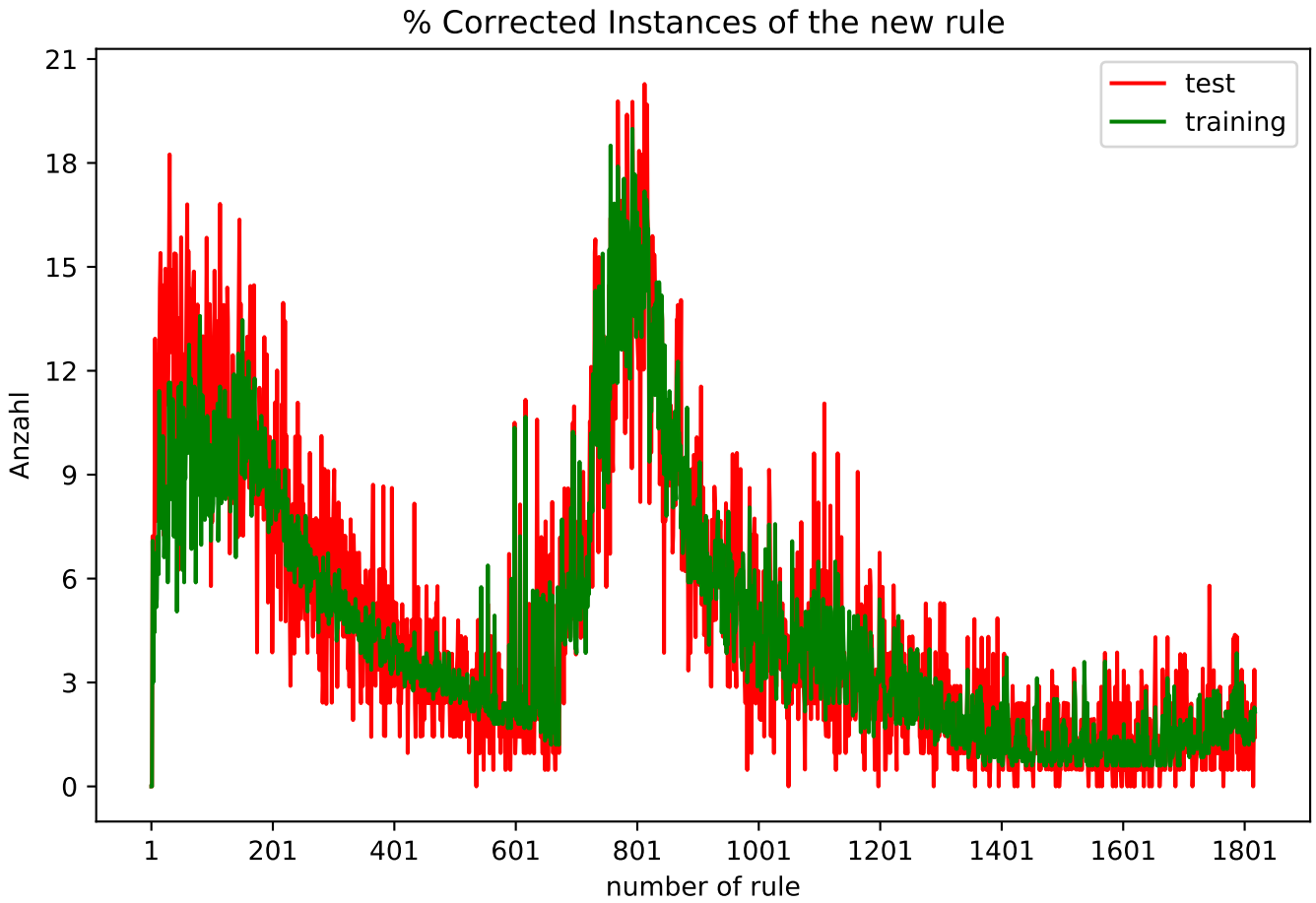

Chosen rules vs rules left
Accuracy
number of rules
model test
model of rules left
84
78
72
66
60
54
48
42
36
30
1
201
401
601
801
1001
1201
1401
1601
1801


Chosen rules vs rules left
Accuracy
number of rules
model trainingdata
model of rules left
96
88
80
72
64
56
48
40
1
201
401
601
801
1001
1201
1401
1601
1801


Compared to random rules
Accuracy
number of rules
test
min/max test
random
min/max random
96
88
80
72
64
56
48
40
1
201
401
601
801
1001
1201
1401
1601
1801


% Covered Instances
Anzahl
number of rules
test
training
96
88
80
72
64
56
48
40
32
1
201
401
601
801
1001
1201
1401
1601
1801


% Covered Instances of the new rule
test
training
Anzahl
number of rule
0
4
8
12
16
20
24
28
32
36
1
201
401
601
801
1001
1201
1401
1601
1801


% Corrected Instances of the new rule
test
training
Anzahl
number of rule
0
3
6
9
12
15
18
21
1
201
401
601
801
1001
1201
1401
1601
1801

sonar.arff-bestrules.csv

Number of rules: 1818.0

Number of traininginstances: 832.0 Number of testinstances: 208.0

Accuracy of the random forest: 85.1

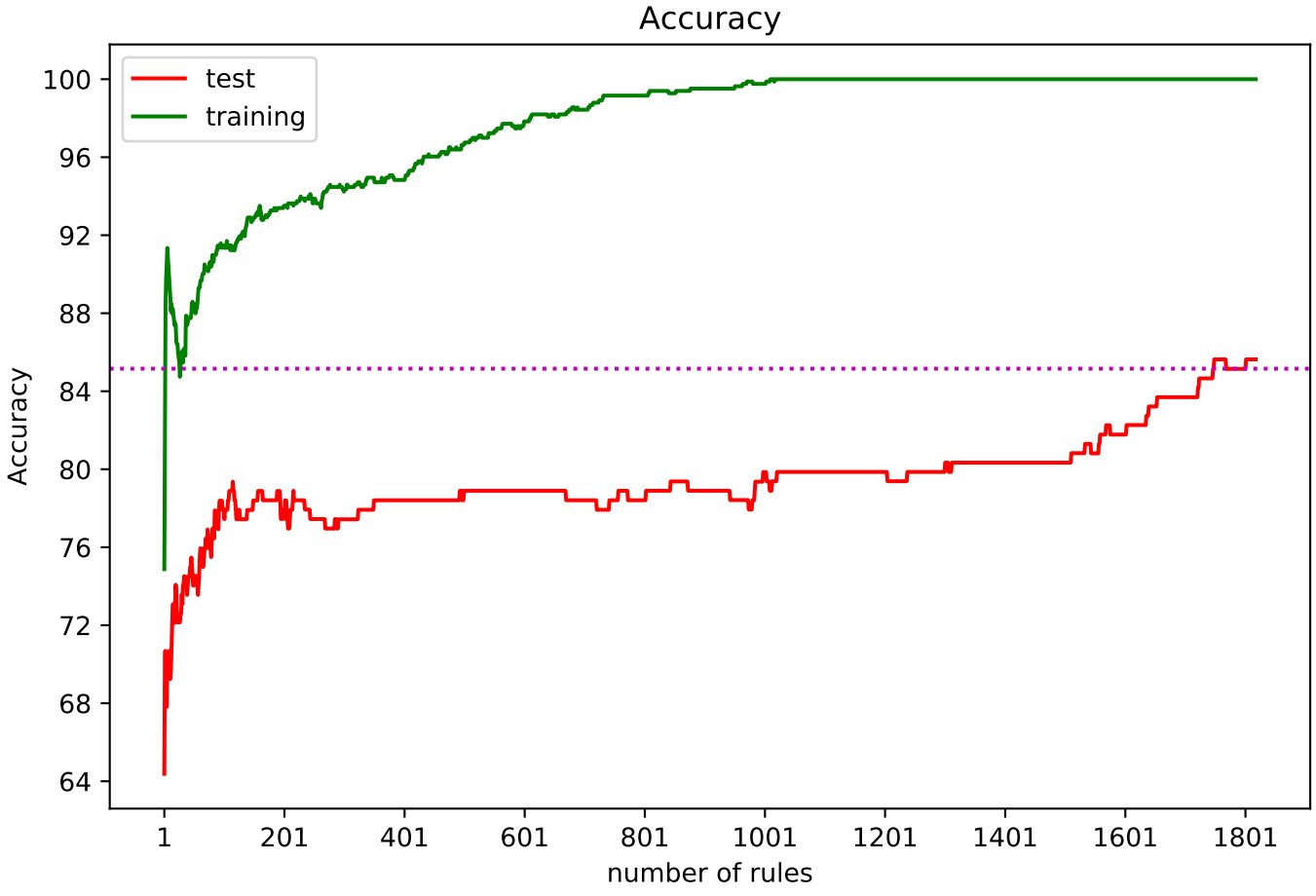
Score integral: 0.76 Score linear: 0.78 Score dcg: 0.76

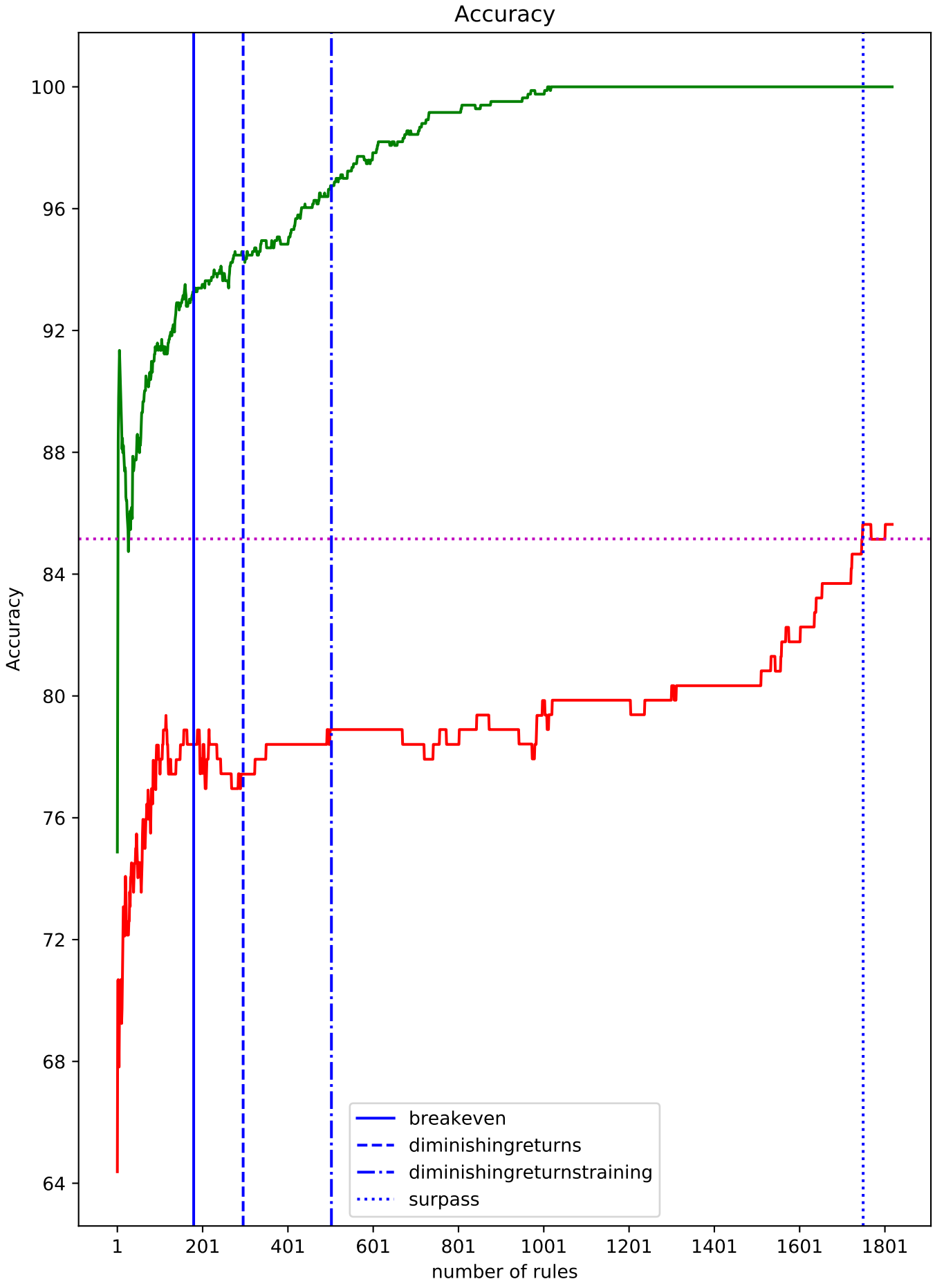
Breakeven: 179.0 rules (Accuracy: 78.4)

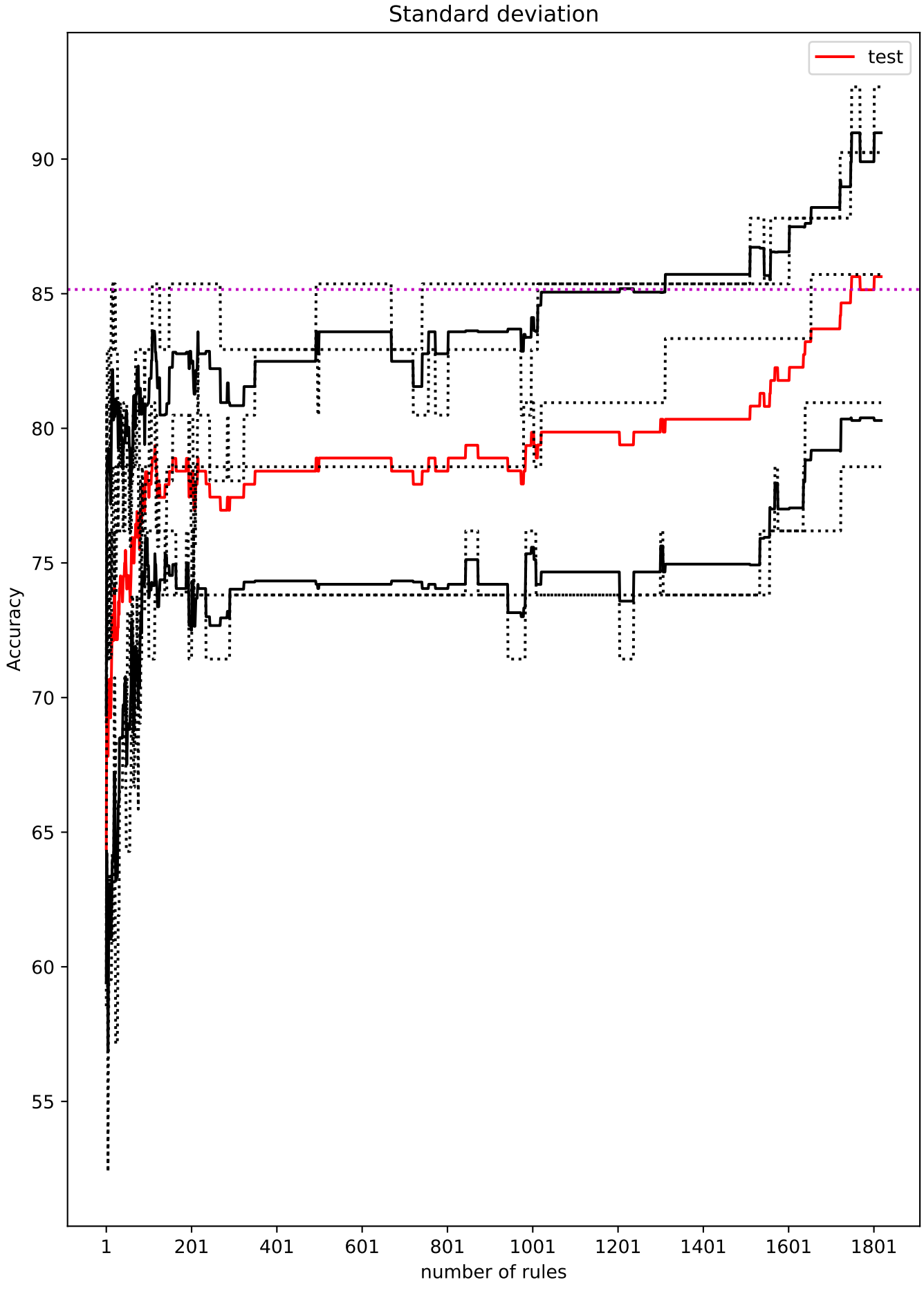
Diminishing returns: 295.0 rules (Accuracy: 77.4)

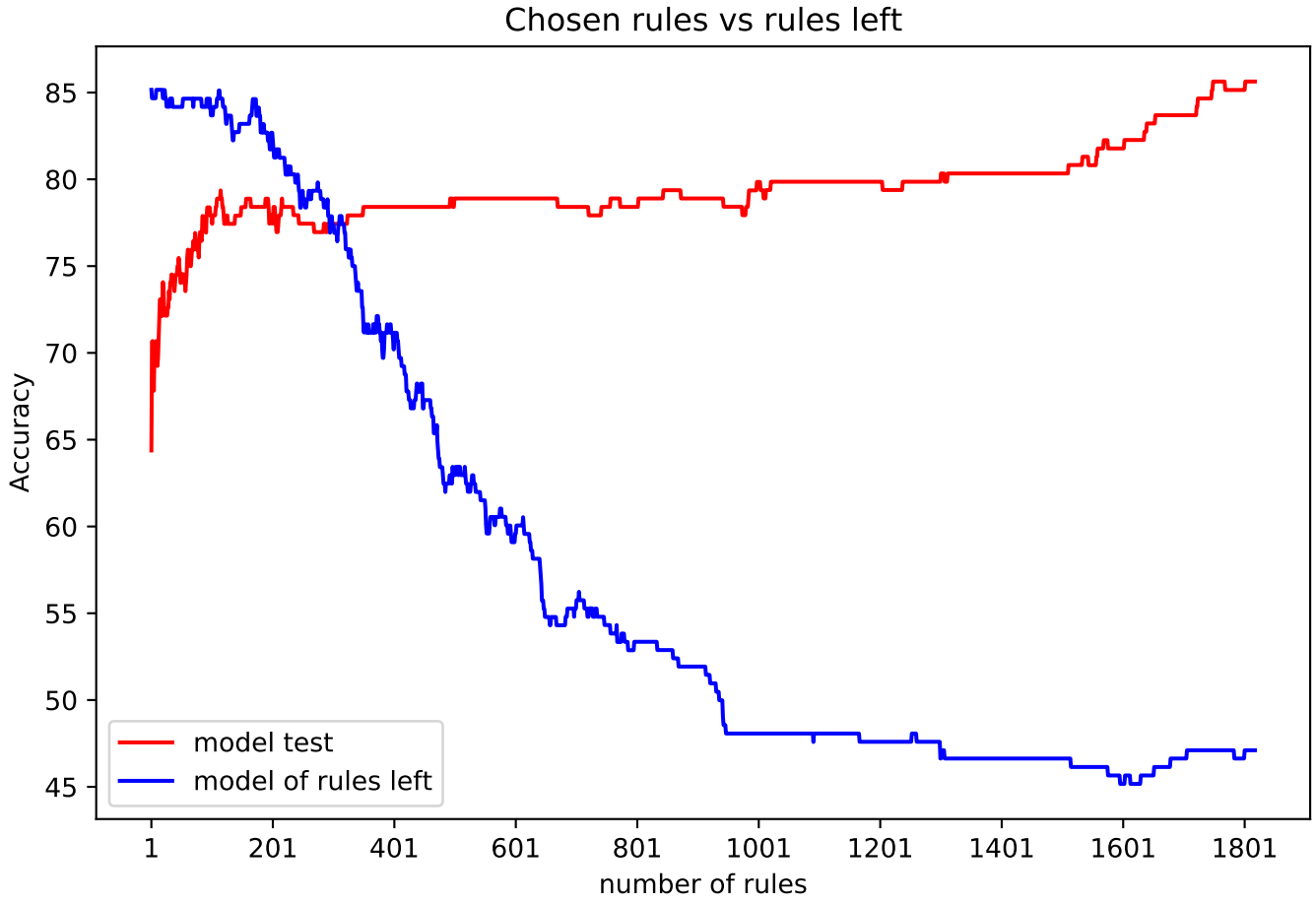
Diminishing returns on trainingdata: 502.0 rules (Accuracy: 78.8)

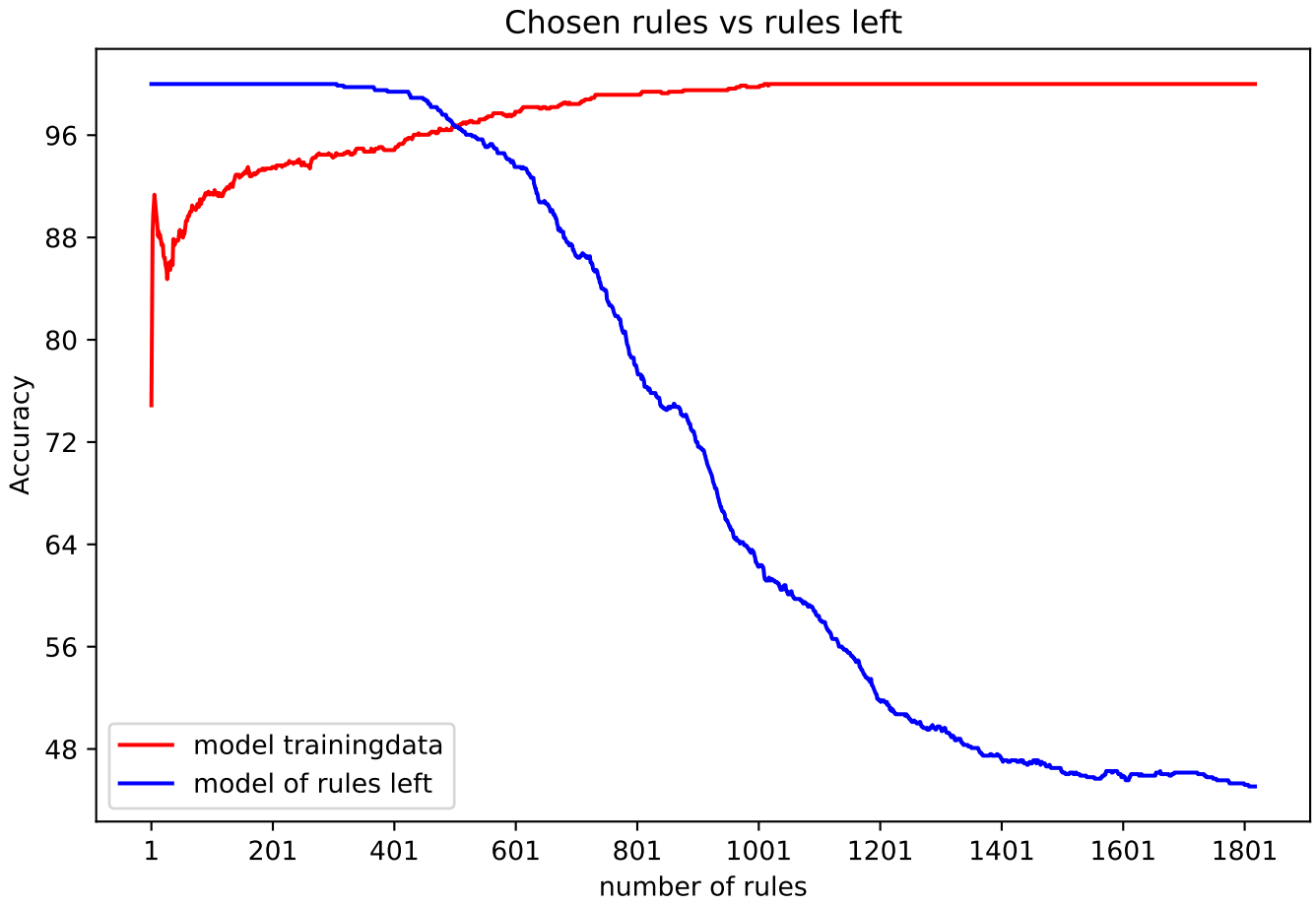
Surpass random forest: 1749.0 rules

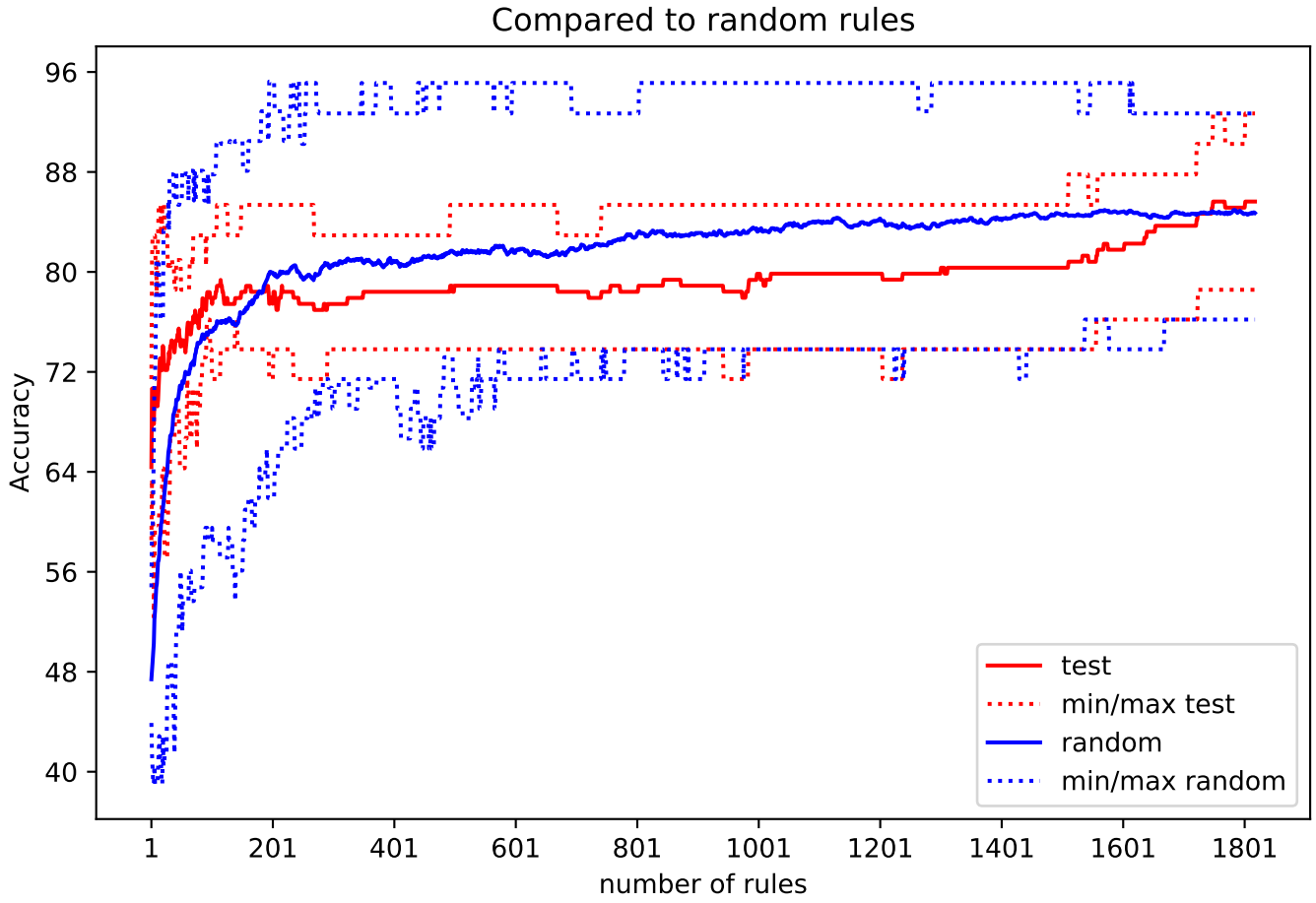


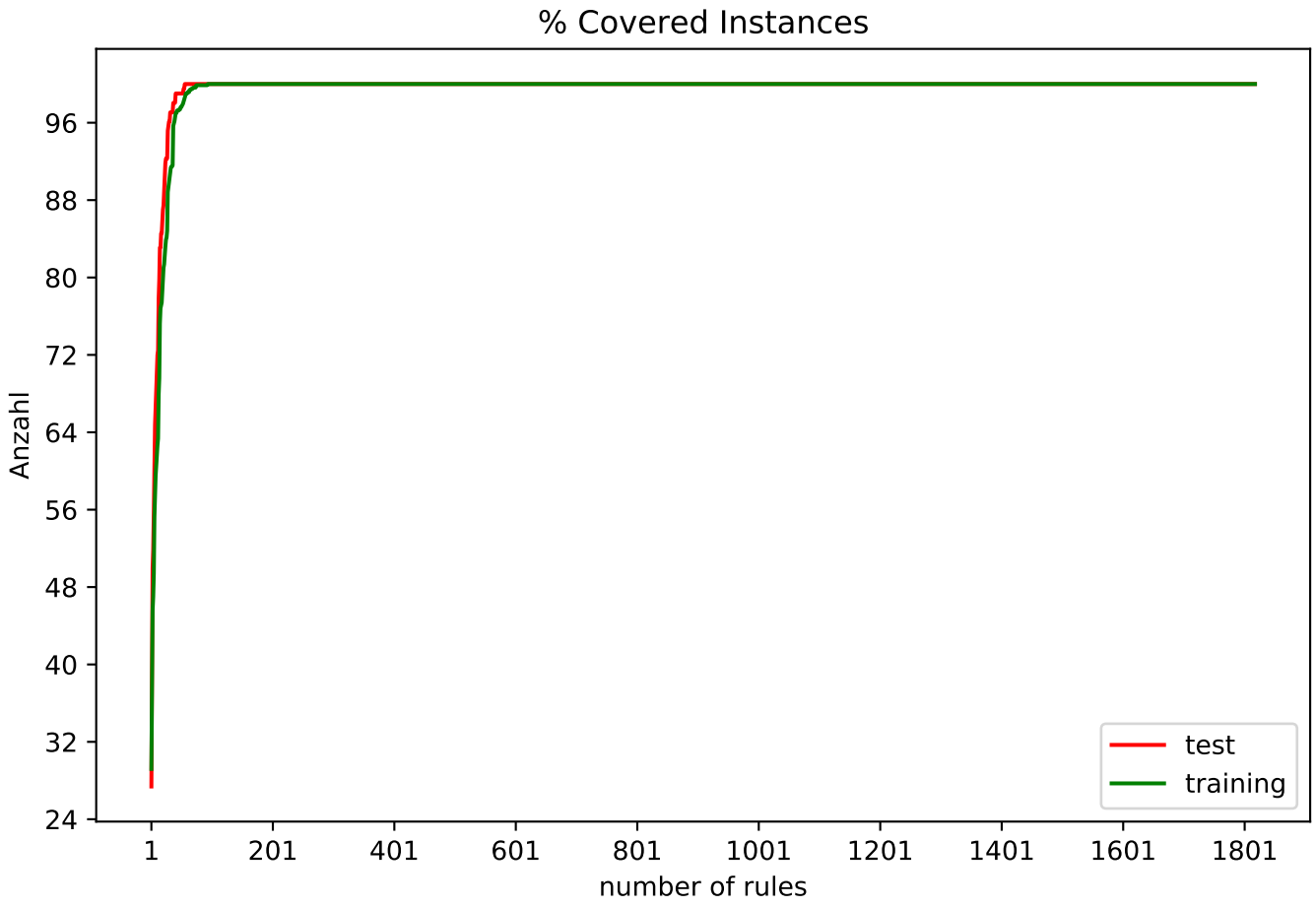

Accuracy
100
96
92
88
84
80
76
72
68
64
Accuracy
1
201
401
601
801
1001
1201
1401
1601
1801
number of rules
breakeven
diminishingreturns
diminishingreturnstraining
surpass

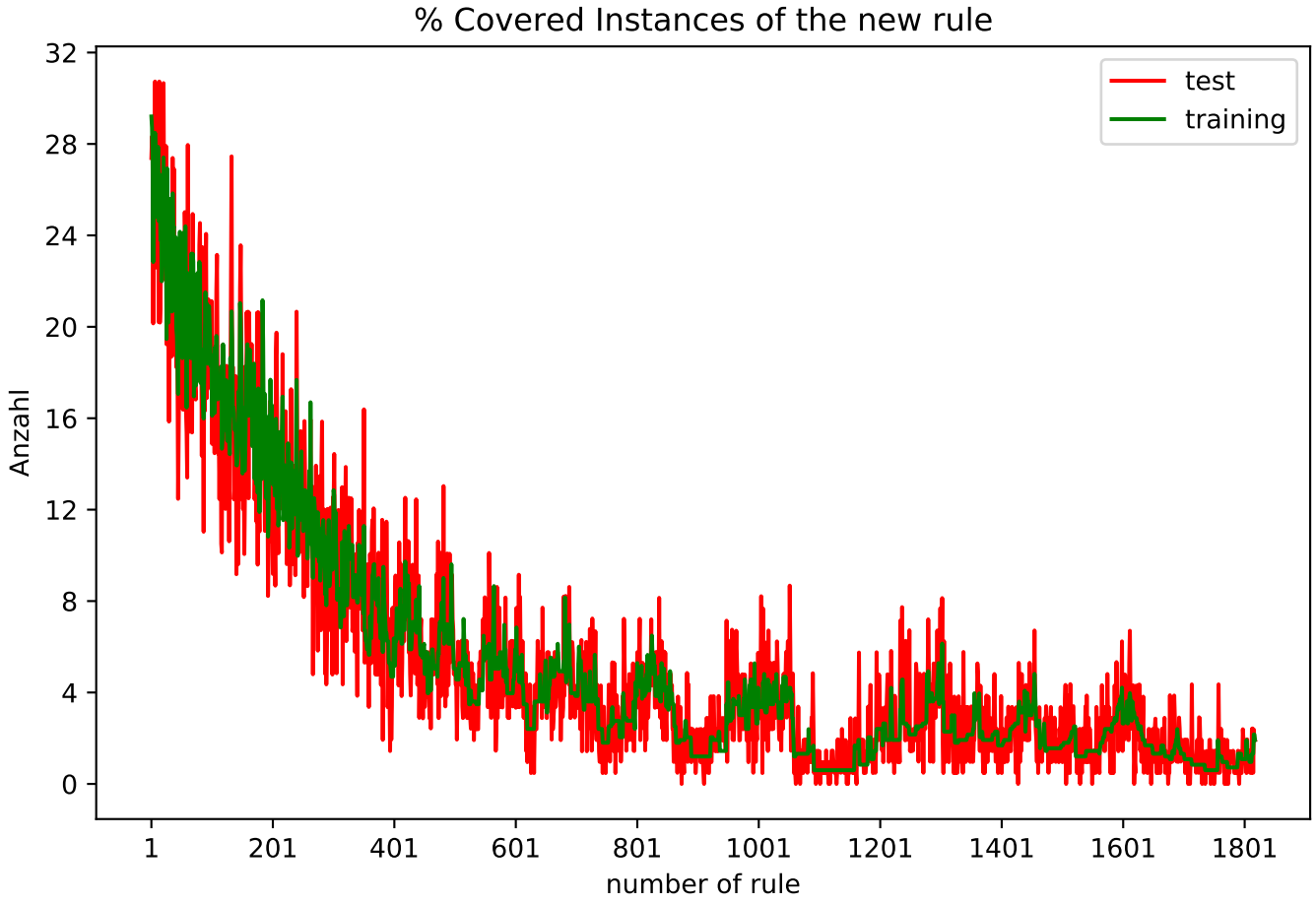

Standard deviation
test
Accuracy
number of rules
90
85
80
75
70
65
60
55
1
201
401
601
801
1001
1201
1401
1601
1801

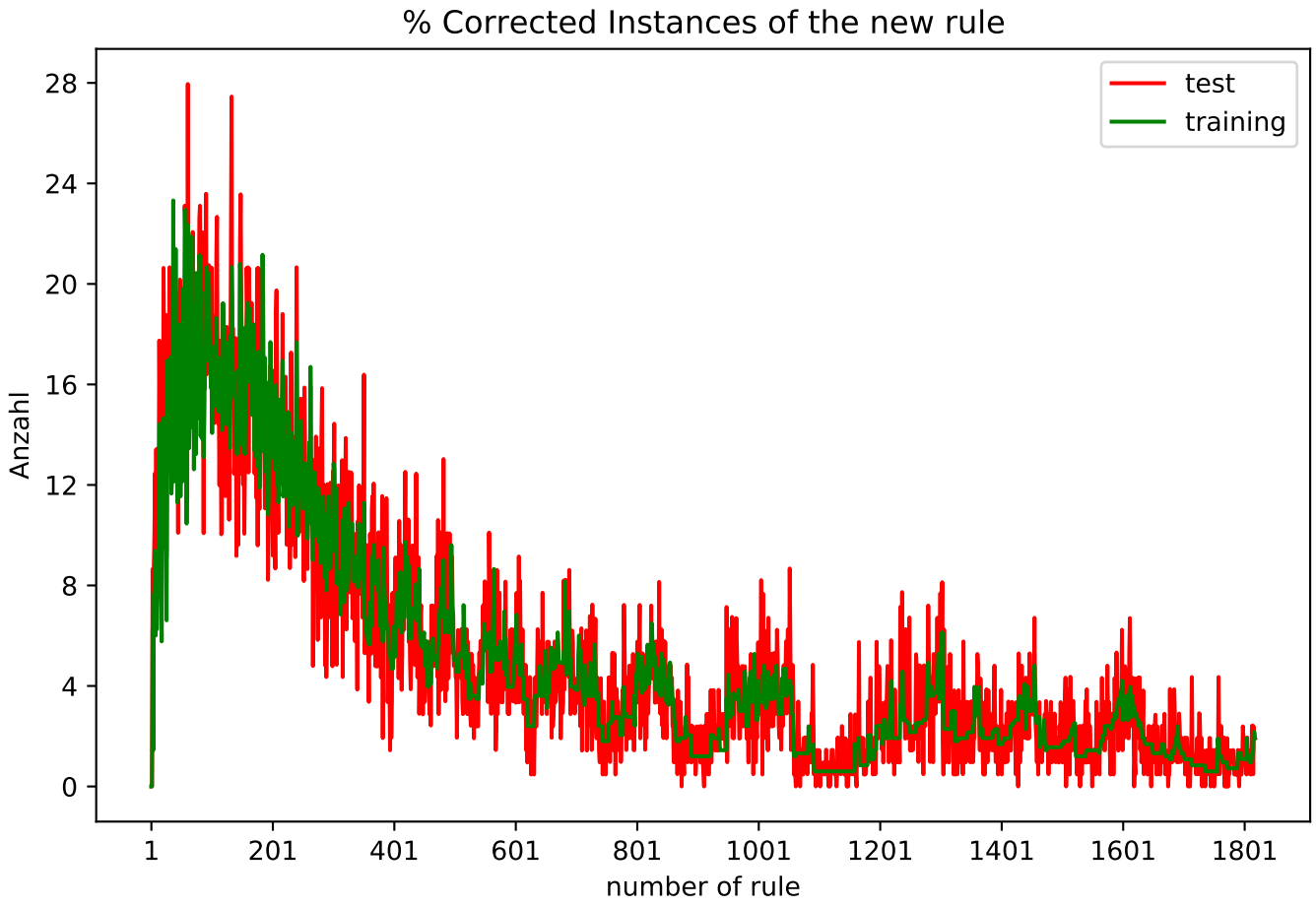

Chosen rules vs rules left
Accuracy
number of rules
model test
model of rules left
85
80
75
70
65
60
55
50
45
1
201
401
601
801
1001
1201
1401
1601
1801


Chosen rules vs rules left
Accuracy
number of rules
model trainingdata
model of rules left
96
88
80
72
64
56
48
1
201
401
601
801
1001
1201
1401
1601
1801


Compared to random rules
Accuracy
number of rules
test
min/max test
random
min/max random
96
88
80
72
64
56
48
40
1
201
401
601
801
1001
1201
1401
1601
1801


% Covered Instances
Anzahl
number of rules
test
training
96
88
80
72
64
56
48
40
32
24
1
201
401
601
801
1001
1201
1401
1601
1801


% Covered Instances of the new rule
test
training
Anzahl
number of rule
0
4
8
12
16
20
24
28
32
1
201
401
601
801
1001
1201
1401
1601
1801


% Corrected Instances of the new rule
test
training
Anzahl
number of rule
0
4
8
12
16
20
24
28
1
201
401
601
801
1001
1201
1401
1601
1801

References

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

Julian Hatwell, Mohamed Medhat Gaber, and R Muhammad Atif Azad. Chirps: Explaining random forest classification. *Artificial Intelligence Review*, 53:5747–5788, 2020.

Richard Maclin and David Opitz. An empirical evaluation of bagging and boosting. *AAAI/IAAI*, 1997:546–551, 1997.

Clemens Otte. Safe and interpretable machine learning: A methodological review. In *Computational intelligence in intelligent data analysis*, pages 111–122. Springer, 2013.

J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

Michael Rapp, Eneldo Loza Mencía, and Johannes Fürnkranz. Simplifying random forests: On the trade-off between interpretability and accuracy. *arXiv preprint arXiv:1911.04393*, 2019.

Vivian S Silva, André Freitas, and Siegfried Handschuh. On the semantic interpretability of artificial intelligence models. *arXiv preprint arXiv:1907.04105*, 2019.

Simonite. When it comes to gorillas, google photos remains blind. *wired.com*. URL https://www.wired.com/story/when-it-comes-to-gorillas-google-photos-remains-blind/.

Taleb Zouggar Souad and Adla Abdelkader. Pruning of random forests: a diversity-based heuristic measure to simplify a random forest ensemble. *INFOCOMP: Journal of Computer Science*, 18(1), 2019.

Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*, 2017.