Interpertable Multiclass Classifiers

L.W. van Someren, 455031
Erasmus University Rotterdam

Supervisor: dr. M.H. Akyuz
Second Assessor: H. Deng MSc

July 2021

Abstract

Over the years, a large number of state-of-the-art classification methods have been proposed, many however, are "black box" models and therefore lack clear interpretability. Although their classification performance is very good, this lack of interpretability is problematic, since it omits a lot of information. Presenting classification methods in a simple and understandable way, enables humans trust a model, as well as design policy based on the findings of the model. To this end, a number of methods based on rule lists have been proposed. In this paper we will compare two such methods, CLASSY and RCAR+. Both these models seek to successfully classify observations in a clear and interpretable way. Based on empirical experiments we find that CLASSY is the more practical of the two.

Contents

1 Introduction and Problem Description **3**

2 Literature **3**
2.1 Machine learning . . . 3
2.2 Interpretability . . . 4
2.3 Linear and Logistic regression . . . 4
2.4 Rule based classification . . . 4
2.5 Alternative approaches . . . 5
2.6 Minimum Description Length . . . 5

3 Methodology **5**
3.1 CLASSY . . . 5
3.1.1 Probabilistic rule lists . . . 5
3.1.2 Estimation methods . . . 5
3.1.3 Minimum Description Length . . . 6
3.1.4 Algorithm . . . 6
3.2 RCAR+ . . . 7
3.2.1 Rule Mining . . . 7
3.2.2 Model fitting . . . 7
3.3 Methods of comparison . . . 8

4 Empirical Experiments **8**
4.1 Data . . . 8
4.2 Results . . . 9
4.2.1 Method comparison using benchmark data . . . 10
4.2.2 Application: Star classification . . . 12

5 Conclusion **13**

A Code **16**
A.1 Data preprosessing (R) . . . 16
A.2 Experiments CLASSY (Python) . . . 16
A.3 Experiments RCAR+ (R) . . . 17

1 Introduction and Problem Description

In recent years, the use of machine learning for classification has become ever more wide spread. Many sophisticated methods have been proposed (Aly, 2005). Many of these methods however, can be characterised as so-called "black box" methods. That is, their inner mechanics are complex and a lot of calculations and steps are needed to perform a single prediction. Therefore it is often unclear what drives the classification. Hence these methods lack a straightforward interpretation. In certain applications this is undesirable, since the classification process might be of greater interest than its result. For instance when researching corporate processes or government policies that lead to a specific outcome. In these cases, there is a desired outcome. Being able to interpret the decisions made by the classification method is crucial to understanding the forces in play. Based on this information, for example, one can better design employee training programs or accurately improve a production process. Rule based algorithms are a solution to this problem, since they are easy to interpret and understand. They do, however, have certain limitations. A rule list of length k has the following form:

Rule		Case		Class probability	Usage
1	**IF**	$\alpha = TRUE$	**THEN**	P(A) = 0.4 P(B) = 0.6	a
2	**ELSE IF**	$\beta = FALSE$	**THEN**	P(C) = 0.5 P(D) = 0.5	b
⋮					
k	**ELSE**		**THEN**	P(Z) = 1.0	z

Here each line corresponds to a rule. For instance if variable α is TRUE then the probability that an observation corresponds to class A is 0.4. The last rule is the default rule, that is, it classifies the observations not assigned by a previous rule. The usage refers to the number of times a rule is used to classify an observation. Most classical rule based classification methods, such as *Repeated Incremental Pruning to Produce Error Reduction* (RIPPER) (Cohen, 1995), have been designed to perform binary classification. They are inapt to deal with the increased complexity encountered in multi-class classification. Since many real world problems have more than 2 classes, new methods were needed. In this paper we will consider two such methods.
First, we will consider the newly proposed CLASSY algorithm (Proença and van Leeuwen, 2020). This method makes use of the *Minimum Description Length* (MDL) principle to extract accurate and concise rule lists from the data. Its benefits include minimal dependency on hyper parameters, hence extensive tuning is not required. This in turn mitigates the risk of overfitting the model. Secondly, we will consider RCAR+ (Azmi and Berrado, 2020). This method seeks to achieve simillar results as the aforementioned CLASSY, but does so in a very different way. The first part of the methods is very similar. When fitting the model however, RCAR+ makes use of multinomial regression to determine which rules should be included. The purpose of this paper is to evaluate the relative performance of these two novel approaches. This will help businesses and policymakers determine the best (and more interpretable) approach currently available, as well as give insight into more promising directions for future research. In the next section we will discuss relevant literature concerning our methods, such as the principles and ideas behind them, as well as alternative approaches. We will then describe their workings and introduce the methods of comparison we will use in our experiments. Next we will discuss data and results for the empirical experiments. Finally we will present our conclusion and considerations for future research.

2 Literature

In this section we will give a brief overview of previous work regarding interpretable learning models, as well as methods to provide interpretation to other non-interpretable classification methods. Furthermore, an introduction to the MDL-principle is provided, as well as a brief history on regression.

2.1 Machine learning

Machine learning models are algorithms that seek to estimate, predict or classify observations based on characteristics (Molnar, 2019). There are two general types of machine learning methods, supervised and unsupervised. In unsupervised learning, we can differentiate different problems between clustering and association. For these problems, method is not trained to give certain responses, but analyses the data for common patterns. These models are often used to gain valuable insight into the data, such as the presence of a clustering structure. A common unsupervised method is K-means-clustering (Friedman et al., 2001). In contrast, supervised learning

makes use of labelled data. There is a known dependent variable, and the model is "trained" to give certain responses using a portion of the data using this variable. Problems that can be solved with supervised learning are classification and prediction. Common supervised learning methods include Linear and Logistic regression, as well as all classification methods discussed in the remainder of this paper.

2.2 Interpretability

As stated in the problem description, the interpretability of classification methods is very important for various reasons. There is, however, no mathematical definition of model interpretability (Molnar, 2019). We therefore must make use of a non-mathematical definition of interpretability to formalise this concept. In Miller, 2019, the authors define interpretability as "*the degree to which a human can understand the cause of a decision*". This definition can successfully be applied to our classification method. When using a *black box* model, such as an Artificial Neural Network (ANN), the inner mechanics of the classifier are unclear. Therefore humans, have trouble understanding the cause of decisions made by such methods. Rule lists, however, provide a clear and concise road map, that can logically classify observations based on specific characteristics. Therefore, humans can understand the cause of decisions made by the algorithm. The importance of interpretability in certain situations cannot be overstated. It has been shown that interpretable models are easier to examine in terms of fairness and causality (Doshi-Velez and Kim, 2017). Furthermore, being able to understand a decision, enables humans to better trust said decision (Ahmad et al., 2018).

2.3 Linear and Logistic regression

Interpretable machine learning models for prediction have been around for a long time. Among the first, and perhaps most famous, was the linear regression model. It was first Legendre and Gauss the early 19th century to predict planetary orbits (Stigler, 1986). A few years later, the method was popularised in the social sciences by Quetelet. Since then, a large number of more advanced methods has been derived from and based on the linear model (Friedman et al., 2001). An adaptation of the linear model for classification is Logistic Regression. The origins of the logistic function, on which the Multinomial Logistic Regression is based, can be traced back to the 1800s (Cramer, 2002). Its first application in the context of regression and binary classification was around in the late 1950s when it was first used to as an alternative to the PROBIT model, which makes use of the cumulative distribution (CDF) of a normal distribution. A decade later, the logistic regression model was extended to multiclass classification by Theil, 1969. Since then it has been used in countless applications (Kleinbaum et al., 2002). It is especially useful in situations where model coefficients need to be analysed. This can be done by analysing the effect of regressors on the odds-ratio's of different classes. This can be interpreted as the effect a regressor has on the probability of being in a certain class. For the purpose of this paper, we will make use of the Lasso regularised regression algorithm (Friedman et al., 2010) to fit our RCAR+ model to the rule space. This regularisation gives explicit control over the model complexity to avoid overfitting.

2.4 Rule based classification

Interpretable classification algorithms based on rule list have been used since the late 1970's (Quinlan, 1979). They have the key advantage of interpretability over so "black box" methods (Fürnkranz et al., 2012). For a long time, rule based classification has been limited to binary problems. This is due to the fact that learning the classifier becomes significantly more complex as the number of classes increases. Early methods for binary classification using probabilistic rule lists, based on greedy heuristics include, among others, CN2 (Clark and Niblett, 1989), Desicion List (Rivest, 1987) and FOIL (Quinlan and Cameron-Jones, 1995). Of these early methods (Cohen, 1995), RIPPER has proved to be the most popular and successful. There exist many methods to generalize RIPPER to multiclass problems. These include, among others, RipMC (Asadi and Shahrabi, 2016) and Jrip (Shahzad et al., 2013). In the late nineties, several methods based on pattern mining, like CBA (Liu et al., 1998) were introduced. These methods first find common patterns in the data using a rule mining algorithm, and then make use of an optimisation techniques to prune these sets as much as possible, while retaining classification accuracy. The most widespread rule miners are APRIORI (Agrawal et al., 1993), which we will also use, and ECLAT (Zaki et al., 1997). The methods used to prune the mined rules are specific to each algorithm. Newer methods based on association rules techniques are BRL (Letham et al., 2015) and IDS (Lakkaraju et al., 2016). Methods based on decision trees, which are closely related to rule lists, have also been developed. Successful classic decision tree methods include CART (Breiman et al., 1984) and C4.5 (QUINLAN, 1993). Decison tree based models sufferer, however, from instability with respect to disturbances in the data (Oates and Jensen, 1997). As a result, Random Forest was developed (Breiman, 2001). It aims to reduce variance in decision trees and thereby increase stability. Methods

using linear programming techniques to construct rule lists have also been proposed (Akyüz and Birbil, 2021). Methods using fuzzy rule sets have also been proved to be successful (Alcala-Fdez et al., 2011). Instead of using thresholds and hard cuts, they make use linguistic terms in their rules to define the antecedents (Ishibuchi et al., 2004). This makes interpretation easy. A major shortcoming is, however, the lack of probability distributions over the classes. Since deterministic classification is sometimes misleading, especially when there are multiple classes, where some may share characteristics and therefore are more suited to probabilistic classification.

2.5 Alternative approaches

To make classification methods interpretable, some have looked beyond rule based models. There exist several methods that seek to explain the inner mechanics of "black box" models. Among the first such models to be used is LIME (Ribeiro et al., 2016). It seeks to explain individual classifications made by the model with an local approximation using a linear model. Since then, several methods have improved upon its shortcomings. An example of which is Anchors (Ribeiro et al., 2018). A major shortcoming of these methods is that, while they provide accurate decisions at the local level, these do not generalise to the global perspective (Mi et al., 2020). Therefore they can not be translated to policy or used to make general real-world decisions.

2.6 Minimum Description Length

The MDL-principle can be used to summarise data (P. D. Grünwald and Grunwald, 2007). It is a method for inductive inference and is based on the principal that any *regularity* in the data can be *compressed*. Compress in this sense implies that the data can be described with fewer symbols or less memory. The degree to which the data can be reduced depends on its regularity/randomness. For instance, while an infinite Fibonnachi sequence can be reduced to a simple formula and two initial values, and infinite sequence of coin tosses cannot be reduced. The main concept is that all data lies somewhere between these two extremes- completely predictable vs completely random -and therefore can be compressed to some extent. The concept is an instance of the more general Ocam's Razor (Gao et al., 2000). Several methods seeking to formalize MDL and use it for data compression have been proposed(Budhathoki and Vreeken, 2015 , Vreeken et al., 2011). Furthermore, it has been used in model selection (P. Grünwald, 2000, Hansen and Yu, 2001). In the context of classification, the MDL principle has also previously been applied. In RIPPER and C4.5 it is used as a criterion for pruning the rule set.

3 Methodology

3.1 CLASSY

3.1.1 Probabilistic rule lists

Consider a supervised Boolean dataset D. That is a boolean dataset X with a label Y. An instance x refers to an observation with k binary values such that $x \in X$. A pattern a occurs in an instance x if x satisfies it. That is, if we have that all elements in a are also in observed in instance x, or $a \subseteq x$. A probabilistic rule list (PRL)is an ordered list of k rules. Each of the rules corresponds to a probability distribution on the classes. That is, if an instance is classified by rule r, it is class 1 with probability p_1 etc. A rule is defined as a pair $r_i = (a_i, \theta(a_i))$. Here pattern a_i is the antecedent and $\theta(a_i))$ is the consequent, a probability distribution over the classes. The rules are a list are ordered in the following way:

$$\begin{aligned} \text{rule 1: IF} a_1 \subseteq x \text{ THEN} \quad & y \sim Categorical(\theta_1) \\ \text{rule 2: ELSE IF} a_2 \subseteq x \text{ THEN} \quad & y \sim Categorical(\theta_2) \\ \text{default: ELSE} \quad & y \sim Categorical(\theta_\emptyset) \end{aligned}$$

An instance x is classified by the first rule r_* that corresponds to the pattern in x. If no rule is activated, the instance is classified by the default (final) rule. Once an instance is classified by a rule r_*, it is assigned to the class that has the highest conditional probability, given that the instance is classified by r_*. More formally, $\hat{y} = \arg\max_{c \in C} \theta^c$

3.1.2 Estimation methods

To estimate the parameters θ_i in the model, the we first have to define support and usage. The support S^{a_i} of a pattern a_i refers to the number of times a specific pattern occurs in the training data. The usage U^{a_i} refers

to how often a rule r_i is invoked. To calculate the categorical distribution for each of the rules, we make use of class-conditioned datasets $D^{y=c}$, which simply is defined as a subset of D such that $y = c$ for all $d \in D^{y=c}$. Similarly, the class conditioned usage is computed as the usage over $D^{y=c}$ and denoted by $U^{(a_i,c)}$. Given these concepts we define a smoothed maximum likelihood estimator as follows:

$$\hat{\theta}_i^c = \frac{U^{(a_i,c)} + \epsilon}{U^{a_i} + |C|\epsilon}$$

3.1.3 Minimum Description Length

To select the optimal model, given our parameter estimates. We resort to the MDL principle. This principle essentially seeks the model that is able to describe the data and the model in the shortest possible code. This can be modelled as $L(D, R) = L(Y|X, R)+L(R)$,or: The combined length of the data and the model is equal to the length of the class labels given the attributes and the model plus the model itself. In order to quantify this we need to encode the model and the data. The model is encoded as per Equation 1. Here $L_{\mathbb{N}}(i) = \log k_0 + \log i + \log\log i + \ldots$ and $k_0 \approx 2.865064$. The pattern length $L(a_i)$ is defined as $L(a_i) = L_{\mathbb{N}}(|a_i|) + |a_i| \log |V|$ where V denotes the set of variables.

$$L(R) = L_{\mathbb{N}}(|R|) + \sum_{a_i \in R} L(a_i) \tag{1}$$

The classlabels are encoded using *prequential plugin code*, which implements our estimator and is defined in Equation 2.

$$P_{plugin}(y_i = c|Y_{i-1} := \frac{|\{(y \in Y_{i-1}|y = c\} + \epsilon|}{\sum_{k \in y} |\{(y \in Y_{i-1}|y = c\} + \epsilon|} \tag{2}$$

The encoding of the full data set can be broken down into the encoding of each of its subset. Therefore it is possible to encode the dataset as $L(Y|D, R) = \sum_{a_i \in R} L(Y^{a_i}|D^{a_i}, R)$

3.1.4 Algorithm

To solve the PRL with MDL-formulation, the CLASSY algorithm makes use of a separate-and-conquer greedy search. This search is characterised by local greedy optimisation of sub-problems, which are then combined to form the global solution. In this instance this means that the algorithm iteratively searches for the next best rule to add to the list. The best rule is defined here as the rule that leads to the largest normalised gain in compression of the data. To calculate the Normalised Compression Gain (NCG), we first need to define Absolute Compression Gain (ACG).

$$\Delta L(D, R \oplus r') = L(D, R) - L(D, R \oplus r') \tag{3}$$

It defined as the difference in code length before and after adding rule r to our list R. We then divide the ACG by the usage, the number of times a rule is invoked, to obtain the NGC.

$$\delta L(D, R \oplus r') = \frac{\Delta L(D, R \oplus r')}{U^a} \tag{4}$$

In the algorithm, candidates are rules that are considered for R. They are generated with a pattern mining algorithm, in this case FP-growth (Borgelt, 2003). The $RemoveRedundant()$ method removes all redundant candidates that satisfy the following two properties: 1. if a rule is a strict subset of another rule, 2. and that same rule has lower of equal support, it is redundant and removed.

Input: dataset D, rule candidate set C
Output: multiclass probabilistic rule list *R*

```
C= RemoveRedundancy(C)
R = ∅
repeat
    r = arg max_{∀r'∈C} : L(D, R⊕ r')
    R = R ⊕ r
    UpdateCandidates(D, R, C)
until δL(D, R⊕ r') ≤ 0, ∀ r' ∈ C;
return R
```

Algorithm 1: CLASSY algorithm

3.2 RCAR+

The Regularized Class Association Rules (RCAR+) algorithm is based on association rules. It works by combining three main elements. The first being a method to mine an exhaustive set of Class Association Rules (CARs). This can be done using a number of pattern mining methods (Azmi and Berrado, 2020). For the purpose of this paper we will make use of APRIORI algorithm (Agrawal, Srikant, et al., 1994). Next the model is fitted using a multinomial regularised logistic regression algorithm. In the final stage we optimise the rules and construct the model. The multiclass version of RCAR+, RCAR+ is implemented in R and included in the *arulesCBA*-package (Hahsler et al., 2020). We will now provide a description of RCAR+.

3.2.1 Rule Mining

To mine the CARs, the algorithm makes use of an pattern mining method, APRIORI. The algorithm decomposes all possible item sets in D, from large to small, and iterativaly checks whether a certain set meets the minimum support threshold. Then the algorithm filters this set of candidates with according to a predefined confidence threshold. If this is the case, a rule $A \Rightarrow C$ is generated. An instance$d \in D$ is associated with class C if it satisfies A. That is if we have all elements in A are also in observed in instance d, or $A \subseteq d$.

Input: dataset D, minimum support level s
Output: Rule set R
$L_1 =$ {size 1-itemset}
for $k = 2, L_{k-1} \neq \emptyset, k++$ **do**
  $C_k = generateCandidates(L_{k-1})$
  foreach $t \in D$ **do**
    $C_t = subset(C_k, t)$
    foreach $c \in C_t$ **do**
      c.count++
    end
    $L_k = \{c \in C_k | c.count \geq s\}$
    $R = R \cup L_k$
  end
end
return R

Algorithm 2: APRIORI algorithm as described in Agrawal, Srikant, et al., 1994

3.2.2 Model fitting

We will now fit the model to the rule set using Multinomial Logistic Regression. Consider the training data D, with N instances and Pattributes. Each instance is associated a class $y \in C$. The set of Q different CARs, mined by the algorithm, which satisfy the support s and confidence c thresholds are defined as follows $R_{s,c} = \{r_q : A_q \Rightarrow y_q\}_{q=1}^{Q}$ with $y_q \in C$. The rule space is given by the transformation in as follows:

$$\mathbb{R}^P \longrightarrow \mathbb{R}^Q$$
$$D \longrightarrow A(D) = \{A_q k(D)\}_{q=1}^{Q}$$

Here A_q represents the condition of rule r_q with the following definion:

$$A_q = \begin{cases} 1 & \text{if } A_q \text{ applies for instance } D_i \\ 0 & \text{otherwise} \end{cases}$$

Note that the probability of case k corresponding to instance i in the rule space $A(D_i)$ can be modeled by means of the multinomial logistic model. Then $\pi_k(A(D_i))$ is the probability of case k in the rule space $A(D_i)$ for parameter $\{\beta_{jk}\}_{j=1}^{Q}$ in the model:

$$\pi_k(A(D_i)) = \frac{\exp(\beta_{0k} + \sum_{q=1}^{Q} \beta_{qk} A_q(D_i))}{\sum_{k=1}^{C} \exp(\beta_{0k} + \sum_{q=1}^{Q} \beta_{qk} A_q(D_i))} \tag{5}$$

The model then fits each rule to a class with the following Lasso penalised log-likelihood function:

$$\{\beta_{jk}\}_{j=1}^{Q} = \max_{(\beta_{0k}, \beta_k) \in Q+1} \frac{1}{N} \sum_{i=1}^{N} (\log P(Y = k | D_i) - \lambda \sum_{q=1}^{Q} |\beta_q| \tag{6}$$

3.3 Methods of comparison

In this section, the ways we compare our two methods are described. To measure accuracy, we will make use of hit-rates. These are computed as the number of correctly predicted instances, divided by the number of incorrectly predicted instances. Furthermore, we will make use of the weighted Recall and weighted Precision measures, as well as the weighted F1-score (Goutte and Gaussier, 2005). In binary classification, Precision is defined as the number of correctly classified positives (True Positives, or TP) divided by the total number of actual positives (TP + FP) in the data. For multi class data sets, we first calculate the precision per class. That is, the proportion of the class that is correctly identified, divided by the total class size. To calculate the weighted precision, we then take the weighted average of the class precision (Sokolova and Lapalme, 2009). Similarly, recall is defined as the proportion of true positives to the total number of predicted positives (TP + FN). We calculate the weighted recall in the same way we do weighted precision. That is, we compute the score per class and then take the weighted average.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

In the example below we calculate the weighted precision (AP) for 3 classes, A, B and C. Here $TP(A)$ denotes true predictions of A, and $\#A$ denotes the actual number of A.

$$WeightedPrecision = \frac{1}{\#A + \#B + \#C}\left(\frac{\#A \times TP(A)}{\#A - TP(A)} + \frac{\#B \times TP(B)}{\#A - TP(B)} + \frac{\#C \times TP(C)}{\#C - TP(C)}\right)$$

Precision and recall cannot be maximised simultaneously, since they imply different things. Maximising precision is done, by making sure there are as little false positives (FP) predictions as possible. This implies a conservative model suitable for situations where the consequences of a positive classification are high, and one needs to be sure the classification is correct. A practical example would be fraud detection. Here it is important to catch fraudulent people. But it is even more important to avoid false positives, due to the severe personal consequences of being marked as a fraud. Avoiding FP-instances comes at the cost of false negative (FN) instances. In other situations, the consequences of an FP classification may be much lower than a FN classification. An example of this is disease detection. Someone who tests false positive may have to quarantine for a few weeks, while someone who is FN can cause an outbreak. Hence all classification problems are a trade off between precision and recall. To evaluate models in an objective way, we need to capture both these traits in a single measure. The F1 score is an example of one such measure. It is defined as the harmonic mean of precision (P) and recall (R).

$$F1 = \frac{2 \times P \times R}{P + R}$$

Furthermore we will make use of *Area Under the ROC Curve* (AUC). The receiver operator characteristic (ROC) curve, plots the TP-rate against the FP-rate for different classification thresholds (Fan et al., 2006). The area under this curve, AUC, can be interpreted as the probability that a random positive instance is ranked higher than a random negative instance (Huang and Ling, 2005). Therefore it serves as a measure of model quality. The AUC measure can be generalised to multiclass classification by averaging one-against-the-rest comparisons of the classes (Hand and Till, 2001). The interpretability will be measured based on the number of rules in the model. Finally we include the run times to indicate computational intensity. This serves as a measure of the model's practicality and scalability.

4 Empirical Experiments

4.1 Data

We will make use of two kinds of data for our experiments. First we will compare our models using 17 widely used benchmark data sets. These were obtained from the LUCS/KDD Repository (Coenen, 2011), and can be found in Table 1. They are distinct in their characteristics in terms of number of instances, variables and class labels. Therefore they will provide a comprehensive and nuanced insight into the method's respective performances. The data sets from this repository are available as index matrices. That is, they have previously been post processed and have been saved as compressed binary matrices. The interpretation of this compression is as follows: A row in the index matrix $[2, 5, 14, 21]$ corresponds to a row in the binary matrix that is zero everywhere, except at index 2, 5, 14 and 21, where it has value 1. In order to run our methods, we reconstruct the binary matrix and the multi class labels. An in-depth description of each data sets is included in the repository.[1]

[1] https://cgi.csc.liv.ac.uk/ frans/KDD/Software/LUCS-KDD-DN/exmpleDNnotes.html

Dataset	Instances	Boolean Variables	Classes
hepatitis	155	48	2
ionosphere	351	155	2
horsecolic	368	81	2
cylBands	540	120	2
breast	699	14	2
pima	768	34	2
tictactoe	958	26	2
mushroom	8124	84	2
adult	48842	96	2
iris	150	14	3
wine	178	63	3
waveform	5000	96	3
heart	303	46	5
pageblocks	5473	39	5
led7	3200	22	10
pendigits	10992	81	10
chessbig	28056	54	18

Table 1: *Dataset overview*

Secondly, we will apply our methods to a real world application. We will seek to classify types of stars based on characteristics like temperature, color, radius. The dataset has been gathered by the North American Space Association (NASA) and was publicly available.[2] A summary of the data can be found in Table 2 There are a total of 6 types, all of which occur in the data 40 times. These are, in order from small to large (by mass):

- Red Dwarf
- Brown Dwarf
- White Dwarf
- Main Sequence
- Super Giant
- Hyper Giant

	Temperature	Luminoscity	Radius	Magnitude	Color		Spectral Class	
Min	1.939	0	0,0084	-11,920	Blue	56	A	19
Max	40.000	849.420	1948,5000	20,060	Blue White	41	B	46
Mean	10.497	107.188,4	237,1578	4,382	Red	112	F	17
1st Q	3.344	0	0,1027	-6,232	White	10	G	1
Median	5.776	0,1	0,7625	8,313	Other	21	K	6
3rd Q	15.056	198.050	42,7500	13,697			M	111
							O	40

Table 2: *Summary of NASA Stars dataset, Luminoscity and Radius are relative to the Sun*

4.2 Results

In order to standardise the results for computational performance, all experiments will be performed on a single machine, with a 2,7 GHz Quad-Core Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 random access memory. Since the implementation of the CLASSY algorithm does not support the use of multi-thread computing, this option was also disabled for RCAR+. To ensure a correct representation of the performance estimates, all experiments will be repeated 10 fold using different random training partitions. This helps to avoid outliers and therefore helps to give an accurate representation of the performance. While performing the RCAR+ experiments, we experienced a severe issue. For the larger datasets (WaveForm, PenDigits and ChessBig), the random access memory (RAM) required during the rule pruning phase would exceed the machines capacity of 16GB. To resolve this, the maximum rule length parameter was limited to 3 from its default of 5 for these datasets. Since this reduced the number of possible rules, it allowed the experiments to run without issue. Due to the fact that this is a practical limitation of the method, it was decided that these results should directly be compared to the other results, without altering the parameters for the smaller datasets and CLASSY model.

[2]https://www.kaggle.com/brsdincer/star-type-classification/metadata

4.2.1 Method comparison using benchmark data

We will now consider the empirical results for the aforementioned benchmark data sets. The performance in terms of accuracy is comparable. For the binary data sets CLASSY was slightly more accurate with an average of 0.85 against RCAR+ with 0.82. A two-sided T-test for the differences of this these means results in a P-value of 0.33. Therefore they are not statistically different at the 5% level. For the multiclass datasets, the difference was larger, with CLASSY and RCAR+ having an average weighted accuracy of 0.80 and 0.72, respectively. Again we use the T-test to determine their differences and find a P-value of 0.31. Hence the means for the multiclass data sets are also not significantly different. There are however, differences per data set. Classy performed better in 10 data sets, while RCAR+ was better in the remaining 7. Overall there seems to be very little difference in accuracy between the two methods.

	Binary datasets								
Model	Hepatitis	Ionosphere	HorseC	CylBands	Breast	Pima	TicTacToe	Mushroom	Adult
CLASSY	0.776	0.877	0.918	0.689	0.924	0.779	0.892	1.000	0.844
RCAR+	0.795	0.932	0.859	0.719	0.905	0.724	0.621	0.999	0.843

Table 3: *Accuracy estimates for binary datasets*

	Multiclass datasets							
Model	Iris	Wine	Waveform	Heart	PageBlocks	Led7	PenDigits	ChessBig
CLASSY	0.923	0.834	0.742	0.970	0.884	0.756	0.949	0.313
RCAR+	0.921	1.000	0.797	0.553	0.895	0.717	0.640	0.260

Table 4: *Weighted Accuracy estimates for multiclass datasets*

We now look to the other performance estimates. As we saw with the estimates for accuracy, the two models are very close. For the binary datasets we see that, on average, only the AUC is statistically different at the 5% level. In all cases CLASSY performs slightly better, although this difference is not significant. For the multiclass data sets, the difference is even less. From the P-value estimates from out 2 sided T-test in Table 6 we find that none are significant at the 5% level. For individual data sets, however, larger differences can be observed. The dataset for which the largest difference in performance can be observed is PenDigets. This dataset has 10 classes, and is only surpassed by ChessBig with 18 classes. From the performance estimates we see that CLASSY still performs well for PenDigets, having all performance measures above 0.94. The results for RCAR+ for this dataset are considerably worse, with weighted precision, recall and F1 valued at 0.557, 0.642 and 0.586, respectively. The AUC is better with an estimate of 0.798, but this is substantially surpassed by CLASSY with an AUC of 0.988. When looking at the largest dataset in terms of classes, ChessBig with 18 classes, we see that the performance for both methods is substantially worse when compared to datasets with fewer classes.

	Precision		Recall		F1		AUC	
Dataset	CLASSY	RCAR+	CLASSY	RCAR+	CLASSY	RCAR+	CLASSY	RCAR+
Hepatitis	0.777	0.500	0.766	0.438	0.771	0.467	0.656	0.662
Ionosphere	0.897	0.915	0.877	0.982	0.871	0.947	0.903	0.915
Horsecolic	0.819	0.833	0.818	0.965	0.818	0.894	0.817	0.825
CylBands	0.799	0.815	0.689	0.400	0.634	0.537	0.781	0.669
Breast	0.934	0.989	0.924	0.864	0.925	0.922	0.951	0.923
Pima	0.771	0.728	0.779	0.946	0.767	0.823	0.718	0.602
TicTacToe	0.921	0.621	0.912	1.000	0.910	0.766	0.972	0.817
Mushroom	1.000	0.998	1.000	1.000	1.000	0.999	1.000	0.999
Adult	0.836	0.749	0.844	0.578	0.838	0.653	0.881	0.756
Mean	**0.862**	0.794	**0.845**	0.797	**0.837**	0.779	**0.853**	0.796
P-value	0.16		0,48		0.22		0.04	

Table 5: *Performance Estimates for Binary Datasets*

	Weighted							
	Precision		Recall		F1		AUC	
Dataset	CLASSY	RCAR+	CLASSY	RCAR+	CLASSY	RCAR+	CLASSY	RCAR+
Iris	0.923	0.908	0.921	0.938	0.921	0.913	0.949	0.969
Wine	0.834	1.000	0.822	1.000	0.823	1.000	0.705	1.000
Waveform	0.742	0.796	0.739	0.797	0.739	0.796	0.894	0.828
Heart	0.970	0.382	0.968	0.566	0.968	0.449	0.973	0.660
PageBlocks	0.884	0.803	0.914	0.898	0.898	0.848	0.702	0.515
Led7	0.756	0.728	0.742	0.718	0.745	0.718	0.936	0.816
PenDigits	0.949	0.557	0.949	0.642	0.949	0.586	0.988	0.798
ChessBig	0.222	0.252	0.292	0.259	0.225	0.190	0.715	0.601
Mean	**0.785**	0.678	**0.793**	0.727	**0.784**	0.688	**0.858**	0.773
P-value	0,41		0,58		0,46		0,28	

Table 6: *Performance Estimates for Multiclass Datasets*

We will now asses the interpretability and practicality of our models. First we will asses the average number of rules per for each model. With one exception, for TicTacToe, CLASSY has on average much fewer rules in comparison to RCAR+. In several cases the difference is extreme. For instance for the Hepatitis dataset (Classy 1, RCAR+ 24), the Wine dataset (Classy 2, RCAR+ 48) and the Heart dataset. For multiclass dataset the ratio's are less extreme, but the difference remains substantial. For instance for the Led7 dataset where Classy use 19 rules, and RCAR+ 164. The differences between the models in terms of runtimes are similar to what we observed for the number of rules. In all cases, CLASSY was substantially faster than RCAR+. The closest RCAR+ came to CLASSy was for the PenDigits dataset, when the ratio, with CLASSY as a baseline, was approximately 3.4. Note however, from Tables 4 and 6 that CLASSY significantly outperformed RCAR+ in this case. More commonly, RCAR+ is 10 times closer. This is the case for 11 out of 17 datasets. An extreme case is the Pima dataset where, on average, RCAR+ was 890 times slower. To illustrate this point, we have plotted the the runtime against the cardiality of the candidate set and the variable set. These plots can be found in in Figure 1. Note that the scale of the candidate set is a log scale. We see that the runtime for RCAR+ increases at a much greater rate than for CLASSY, both in terms of total number of variables and candidates. This is problematic for the practicality of RCAR+, as it limits the size of the datasets for which it can be used. The long runtimes for WaveForm, PenDigits and ChessBig (679.22, 621.95 and 897.98, respectively,) are especially notable. This is because of the aforementioned limitation of maximum rule length due to lack of RAM.

	Number of Rules		Runtime (sec.)	
	CLASSY	RCAR+	CLASSY	RCAR+
Hepatitis	1	24	2.33	90.19
Ionosphere	4	42	13.49	606.97
Horsecolic	1	14	1.51	69.32
CylBands	2	36	2.86	353.97
Breast	2	14	0.73	2.95
Pima	2	6	0.22	195.86
TicTacToe	9	8	15.69	89.8
Mushroom	4	32	11.91	87.85
Adult	35	124	38.54	1128.75
Iris	2	29	1.92	13.04
Wine	2	48	2.25	1016.08
Waveform	21	100	69.71	679.22
Heart	1	20	22.99	654.01
PageBlocks	6	23	24.15	297.38
Led7	19	164	23.29	256.94
PenDigits	24	31	187.81	621.95
ChessBig	18	30	23.97	897.98

Table 7: *Interpretability and efficiency estimates for Benchmark Datasets*

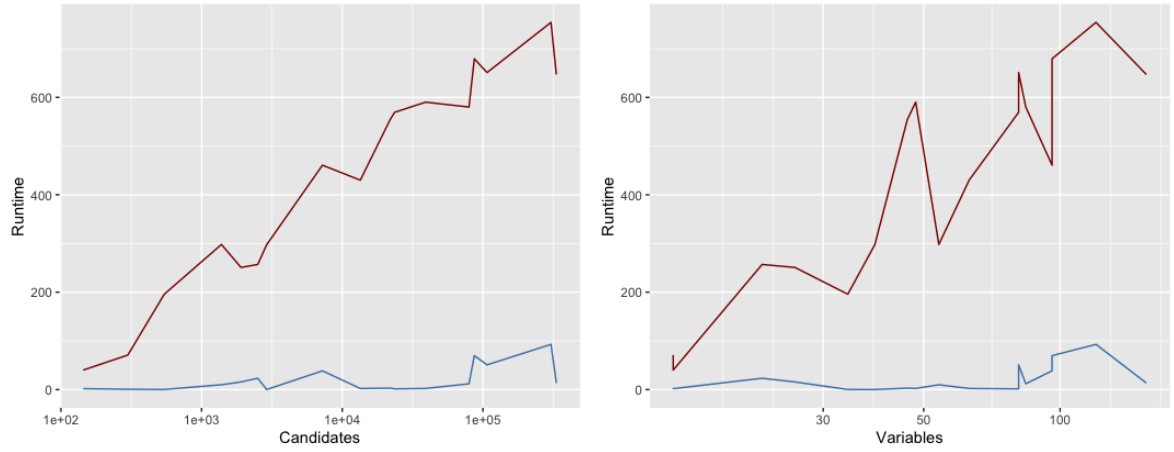

Figure 1: *Runtime versus Candidates and Variables for RCAR+ (red) and CLASSY (blue)*

4.2.2 Application: Star classification

We now turn to the applied experiment where we seek to evaluate the quantitative and qualitative performances of our respective models in classifying stars based on their features. In terms of Classification Performance, we see from Table 8 that CLASSY performs better. Across all measures, CLASSY scores higher than RCAR+. Upon examining the one-against-rest ROC curves for both estimators, found in Figure 2, we see that this is mainly due a low AUC for Class 0 (Red Dwarf). Note that since the ROC curves are averaged across 10 different training sets, it is unlikely this is due to a bad fit. CLASSY clearly performs better on this particular data set.

		Weighted			
Model	Accuracy	Precision	Recall	F1	AUC
CLASSY	0.986	0.987	0.986	0.986	0.991
RCAR+	0.858	0.862	0.858	0.858	0.953

Table 8: *Average Classification Performance for Star Classification*

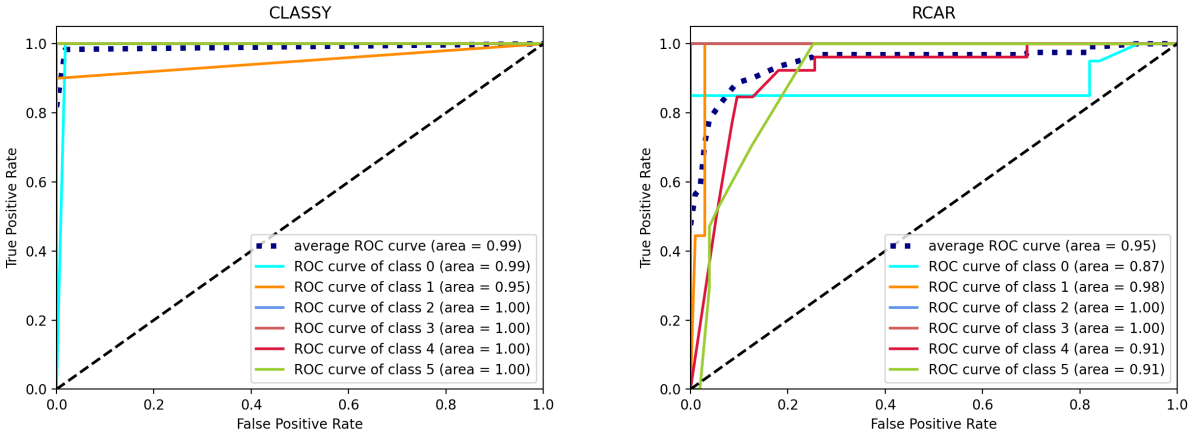

Figure 2: *Average ROC curves for Star classification*

The difference in performance becomes much larger when comparing the interpretability and computational intensity. The average runtime for RCAR+ was 8.74 seconds, compared to just 1.07 seconds for CLASSY. For this data set the difference may seem trivial, but as stated in Section 4.2.1, it becomes problematic for larger datasets. In terms of interpretability, the number of conditions in the RCAR+ model is 4 times as large as the in the CLASSY model.

Model	Rules	Conditions	Average Rule Length	Runtime
CLASSY	6	12	2.00	1.07
RCAR+	26	48	1.85	8.74

Table 9: *Interpretability estimates for Star Classification*

We further investigate the interpretability by directly examining the models. We see that the model trained by the CLASSY algorithm is very useful in practice. With only a single rule per class, the model is easy to interpret by anyone, even those without mathematical knowledge or training. Furthermore, the class distribution gives key insight into the confidence associated with the estimate. A similar measurement of confidence is provided by the RCAR+ model. This is not a complete distribution, however, but a one-versus-the-rest confidence indication. This can be problematic when the rule confidence is low, since no information on competing classes is included.

Star Classification using CLASSY

	Rule	Condition				Class probability						Usage
						RD	BD	WD	MS	SG	HG	
1	**IF**	*Temp* $\geq$ 3572.0	**AND**	*Rad* $<$ 0.09	**THEN**	0.0	0.0	1.0	0.0	0.0	0.0	26
2	**ELSE IF**	*Rad* $<$ 0.24	**AND**	*Mag* $\geq$ 14.36	**THEN**	1.0	0.0	0.0	0.0	0.0	0.0	23
3	**ELSE IF**	*Lum* $<$ 0.72	**AND**	*SC* = M	**THEN**	0.0	1.0	0.0	0.0	0.0	0.0	30
4	**ELSE IF**	*Rad*$<$ 24.5	**AND**	*Mag*$\geq$ -5.87	**THEN**	0.0	0.0	0.0	1.0	0.0	0.0	27
5	**ELSE IF**	*Rad* $<$ 1000.0	**AND**	*Mag*$\geq$ -7.65	**THEN**	0.0	0.0	0.0	0.0	1.0	0.0	31
6	**ELSE**					0.0	0.0	0.0	0.0	0.0	1.0	31

Star Classification using RCAR+

Rule	Condition						Class	Probability
1	**IF**	*Temp* = [1.94e+03 , 3.53e+03)	**AND**	*Rad* = [0.0084,0.158)	**THEN**		RD	0.87
2	**IF**	*Rad* = [0.0084 , 0.158)			**THEN**		RD	0.500
3	**IF**	*Rad* = [0.158 , 11.1)	**AND**	*Color* = Red	**THEN**		BD	1.00
4	**IF**	*Rad* = [0.158 , 11.1)	**AND**	*Mag* = [-4.88,12.4)	**THEN**		MS	0.75
5	**IF**	*Rad* = [0.0084 , 0.158)	**AND**	*Mag* = [12.4,20.1]	**THEN**		RD	0.67
6	**IF**	*Temp* = [1.94e+03 , 3.53e+03)	**AND**	*Mag* = [12.4,20.1]	**THEN**		RD	0.69
7	**IF**	*Lum* = [1.24e+05 , 8.49e+05]	**AND**	*Rad* = [11.1,1.95e+03]	**THEN**		HG	0.58
8	**IF**	*Temp* = [1.94e+03 , 3.53e+03)	**AND**	*Lum* = [8e-05,0.00131)	**THEN**		RD	0.83
9	**IF**	*Rad* = [11.1 , 1.95e+03]	**AND**	*Mag* = [-11.9,-4.88)	**THEN**		HG	0.62
10	**IF**	*Rad* = [11.1 , 1.95e+03]			**THEN**		WD	0.62
11	**IF**	*Rad* = [11.1 , 1.95e+03]	**AND**	*Color* = Red	**THEN**		HG	0.80
12	**IF**	*Lum* = [1.24e+05 , 8.49e+05]	**AND**	*Mag* = [-11.9,-4.88)	**THEN**		HG	0.58
13	**IF**	*Temp* = [3.53e+03 , 1.19e+04)	**AND**	*Rad* = [11.1,1.95e+03]	**THEN**		HG	0.81
14	**IF**	*Color* = Blue-white ,			**THEN**		MS	0.75
15	**IF**	*Rad* = [11.1 , 1.95e+03]	**AND**	*SC* = M	**THEN**		HG	0.80
16	**IF**	*Lum* = [8e-05 , 0.00131)	**AND**	*Color* = Red	**THEN**		RD	0.70
17	**IF**	*Temp* = [1.19e+04 , 4e+04]	**AND**	*Rad* = [0.158,11.1)	**THEN**		MS	1.00
18	**IF**	*Mag* = [-11.9 , -4.88)	**AND**	*SC* = M	**THEN**		HG	0.80
19	**IF**	*Temp* = [3.53e+03 , 1.19e+04)	**AND**	*Mag* = [-11.9,-4.88)	**THEN**		WD	0.81
20	**IF**	*Rad* = [0.0084 , 0.158)	**AND**	*Color* = Red	**THEN**		RD	0.87
21	**IF**	*Rad* = [0.158 , 11.1)	**AND**	*SC* = M	**THEN**		BD	1.00
22	**IF**	*Lum* = [8e-05 , 0.00131)	**AND**	*SC* = M	**THEN**		RD	0.70
23	**IF**	*Mag* = [-11.9 , -4.88)			**THEN**		HG	0.62
24	**IF**	*Lum* = [0.00131 , 1.24e+05)	**AND**	*Color* = Red	**THEN**		BD	0.78
25	**IF**	*Lum* = [0.00131 , 1.24e+05)	**AND**	*SC* = M	**THEN**		BD	0.78
26	**IF**	*Mag* = [-11.9 , -4.88)	**AND**	*Color* = Red	**THEN**		HG	0.80

5 Conclusion

In this paper we set out to evaluate interpretable classification models CLASSY and RCAR+. To this end we performed several empirical experiments. In total we made use of 18 datasets for our comparison. Across almost all data sets, the classification performance of CLASSY and RCAR+ was comparable. In terms of interpretability

and practicality, however, there were more significant distinctions. Without exception, CLASSY had fewer rules and was faster than RCAR+. In addition to the dramatic difference in number of rules, RCAR+ requires substantially computing power for similar or worse results. Therefore we conclude that for the purposes of generating a model that is easy to interpret and use, CLASSY is the obvious choice. Perhaps if an additional phase of pruning on the RCAR+ models was performed, they would be comparable in ease-of-use. This does not, however, address the concerns regarding runtime and computational intensity. This brings us to the limitations of the experiments conducted in this paper. Since both CLASSY and RCAR+ have been implemented by different people, the relative quality of the implementations could skew the results. This is especially relevant for the results regarding runtime as small inefficiencies in the code can lead to significant differences in run time. A possible consideration for future research is to include an analysis of the method implementations, for any inefficiencies. Another consideration is to implement another rule pruning step for the RCAR+ method. This would convert the output format to resemble the consecutive rules produced by CLASSY, as well as remove any redundant rules.

References

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, 207–216.

Agrawal, R., Srikant, R. et al. (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB*, *1215*, 487–499.

Ahmad, M. A., Eckert, C., & Teredesai, A. (2018). Interpretable machine learning in healthcare. *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, 559–560.

Akyüz, M. H., & Birbil, Ş. İ. (2021). Discovering classification rules for interpretable learning with linear programming. *arXiv preprint arXiv:2104.10751*.

Alcala-Fdez, J., Alcala, R., & Herrera, F. (2011). A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. *IEEE Transactions on Fuzzy Systems*, *19*(5), 857–872. https://doi.org/10.1109/TFUZZ.2011.2147794

Aly, M. (2005). Survey on multiclass classification methods. *Neural Netw*, *19*, 1–9.

Asadi, S., & Shahrabi, J. (2016). Ripmc: Ripper for multiclass classification. *Neurocomputing*, *191*, 19–33.

Azmi, M., & Berrado, A. (2020). Rcar framework: Building a regularized class association rules model in a categorical data space. *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*.

Borgelt, C. (2003). Efficient implementations of apriori and eclat. *FIMI'03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*, 90.

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

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Budhathoki, K., & Vreeken, J. (2015). The difference and the norm—characterising similarities and differences between databases. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 206–223.

Clark, P., & Niblett, T. (1989). The cn2 induction algorithm. *Machine learning*, *3*(4), 261–283.

Coenen, F. (2011). The lucs-kdd data discretization/normalization (dn) java software for classification association rule mining version 2. *red*, *25*(56), 1.

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

Cramer, J. S. (2002). The origins of logistic regression.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Fan, J., Upadhye, S., & Worster, A. (2006). Understanding receiver operating characteristic (roc) curves. *Canadian Journal of Emergency Medicine*, *8*(1), 19–20.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2001). *The elements of statistical learning* (Vol. 1). Springer series in statistics New York.

Fürnkranz, J., Gamberger, D., & Lavrač, N. (2012). *Foundations of rule learning*. Springer Science & Business Media.

Gao, Q., Li, M., & Vitányi, P. (2000). Applying mdl to learn best model granularity. *Artificial Intelligence*, *121*(1-2), 1–29.

Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. *European conference on information retrieval*, 345–359.

Grünwald, P. (2000). Model selection based on minimum description length. *Journal of mathematical psychology*, *44*(1), 133–152.

Grünwald, P. D., & Grunwald, A. (2007). *The minimum description length principle*. MIT press.

Hahsler, M., Johnson, I., & Giallanza, T. (2020). Package 'arulescba'.

Hand, D. J., & Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, *45*(2), 171–186.

Hansen, M. H., & Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, *96*(454), 746–774.

Huang, J., & Ling, C. X. (2005). Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, *17*(3), 299–310.

Ishibuchi, H., Nakashima, T., & Nii, M. (2004). *Classification and modeling with linguistic information granules: Advanced approaches to linguistic data mining*. Springer Science & Business Media.

Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., & Klein, M. (2002). *Logistic regression*. Springer.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1675–1684.

Letham, B., Rudin, C., McCormick, T. H., Madigan, D., et al. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, *9*(3), 1350–1371.

Liu, B., Hsu, W., Ma, Y., et al. (1998). Integrating classification and association rule mining. *Kdd*, *98*, 80–86.

Mi, J.-X., Li, A.-D., & Zhou, L.-F. (2020). Review study of interpretation methods for future interpretable machine learning. *IEEE Access*, *8*, 191969–191985. https://doi.org/10.1109/ACCESS.2020.3032756

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, *267*, 1–38.

Molnar, C. (2019). *Interpretable machine learning: A guide for making black box models explainable* [https://christophm.github.io/interpretable-ml-book/].

Oates, T., & Jensen, D. (1997). The e! ects of training set size on decision tree complexity. *Proc. 14th Int. Conf. on Machine Learning*.

Proença, H. M., & van Leeuwen, M. (2020). Interpretable multiclass classification by mdl-based rule lists. *Information Sciences*, *512*, 1372–1393.

QUINLAN, J. R. (1993). Program for machine learning. *C4.5*. https://ci.nii.ac.jp/naid/10015645285/en/

Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. *Expert systems in the micro electronics age*.

Quinlan, J. R., & Cameron-Jones, R. M. (1995). Induction of logic programs: Foil and related systems. *New Generation Computing*, *13*(3-4), 287–312.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Rivest, R. L. (1987). Learning decision lists. *Machine learning*, *2*(3), 229–246.

Shahzad, W., Asad, S., & Khan, M. A. (2013). Feature subset selection using association rule mining and jrip classifier. *International Journal of Physical Sciences*, *8*(18), 885–896.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing Management*, *45*(4), 427–437. https://doi.org/https://doi.org/10.1016/j.ipm.2009.03.002

Stigler, S. M. (1986). *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press.

Theil, H. (1969). A multinomial extension of the linear logit model. *International economic review*, *10*(3), 251–259.

Vreeken, J., Van Leeuwen, M., & Siebes, A. (2011). Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, *23*(1), 169–214.

Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). Parallel algorithms for discovery of association rules. *Data mining and knowledge discovery*, *1*(4), 343–373.

A Code

A.1 Data preprosessing (R)

```
library(readr)
library(phonTools)

input_data <- as.matrix(read_delim("Documents/Thesis/Data/horseColic.csv",
                        "␣", escape_double = FALSE, col_names = FALSE,
                        trim_ws = TRUE))
dim = 85
n_classes = 2
############ Process data #########
my_data = zeros(nrow(input_data),dim)
#create binary matrix from index matrix
for (row in 1:nrow(input_data)) {
  for (column in 1:ncol(input_data)) {
    if (!is.na(input_data[row,column])) {
      colnr = input_data[row,column]
      my_data[row,colnr] = 1
    }
  }
}
#construct (multiclass) label from final c columns
class_indicators = my_data[,(dim-n_classes+1):dim]
my_data = as.data.frame(my_data)[1:(dim-n_classes)]
label = class_indicators[,1]
for (row in 1:nrow(input_data)) {
  for (class in 1:n_classes)
    if (class_indicators[row,class] == 1) {
      label[row] = class
    }
}
y = label
my_data = cbind(y,my_data)
my_data = my_data[my_data$y != 0,]
my_data$y = factor(my_data$y)
rm(input_data)
rm(class_indicators)
rm(y)
########### Export binarized data to CSV

write.csv(my_data, "horseColic.csv")
```

A.2 Experiments CLASSY (Python)

```
from pandas import read_csv
from rulelist import RuleList
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score,
                            precision_score, f1_score, confusion_matrix
import numpy as np

my_data = read_csv("./datasets/hepatitis.csv")

Y = my_data.y
weights = Y.value_counts(sort=False)
X = my_data.drop('y', axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25
                                                    , random_state=94)

model = RuleList(task='prediction', target_model='categorical'
                                 , min_support=0.1, beam_width=100, max_depth=5)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

y_prob = model.predict_proba(X_test)
y_prob = np.vstack((y_train.unique(), y_prob))
y_prob = y_prob[:, y_prob[0].argsort()]
y_prob = np.delete(y_prob, 0, 0)

if y_test.unique().size == 2:
    y_prob = y_prob[:, 1]
    binary_prediction = y_pred

accuracy = str(round(accuracy_score(y_test.values, y_pred), 3))
recall = str(round(recall_score(y_test.values, y_pred, average="weighted"), 3))
precision = str(round(precision_score(y_test.values, y_pred, average="weighted")
                , 3))
f1_score = str(round(f1_score(y_test.values, y_pred, average="weighted"), 3))
AUC = str(round(roc_auc_score(y_test, y_prob, average="weighted", multi_class="ovr"), 3))

num_rules = str(model.number_rules)
runtime = round(model.runtime, 3)
print("%s␣%s␣%s␣%s␣%s␣%s␣%s" %
(num_rules, runtime, accuracy, precision, recall, f1_score, AUC))
```

A.3 Experiments RCAR+ (R)

```
library(arulesCBA)
library(caTools)
library(pROC)
library(caret)
library(e1071)
library(mlbench)
library(kmed)
library(tictoc)
library(evclass)
library(readr)
library(phonTools)

my_data <- read_csv("Documents/Thesis/datasets/iris.csv")
my_data$X1 = NULL
my_data$y = factor(my_data$y)


########### Partition data #######
smp_size <- floor(0.75 * nrow(my_data))
set.seed(125)

train_ind <- sample(seq_len(nrow(my_data)), size = smp_size)

train <- my_data[train_ind, ]
test <- my_data[-train_ind, ]
```

```
result =test[1]
colnames(result)  = c("actual")
test <- test[-1]
############## Estimate RCAR+  + runtime ##########
tic("Model_estimate")
RCAR+_model = RCAR+(y~., data = train, verbose = TRUE ,cv.glmnet.args = list(nfolds = 10),
                    parameter = list(confidence = 0.9, support = 0.1, minlen=1,  maxlen = 5)
                    lambda = NULL )
runtime = toc()

######### Calculate hitrate #######
result$prediction = predict(RCAR+_model, test)
accuracy = (sum(result[1] == result[2]))/nrow(result)

############ Calculate num Rules #########
weights_rule = RCAR+_model[["rules"]]@quality[["weight"]]
Num_rules = length(weights_rule[weights_rule > 0.05])
########### Confusion Matrix & scores #############
matrix = confusionMatrix(result$prediction,result$actual)
my_mat = matrix$byClass
num_classes = nlevels(result$actual)
if (num_classes  > 2) {
  classweights = table(my_data$y)

  my_mat[is.na(my_mat)] = 0
  Weighted_F1 = sum(my_mat[,7] * classweights)/sum(classweights)
  Weighted_recall =  sum(my_mat[,6] * classweights)/sum(classweights)
  Weighted_precision =  sum(my_mat[,5] * classweights)/sum(classweights)
}else{
  Weighted_F1 = my_mat[7]
  Weighted_recall =  my_mat[6]
  Weighted_precision =my_mat[5]
}

############# Calculate AUC #############
AUC_result = multiclass.roc( unclass(result$actual),unclass(result$prediction))$auc

############ Write results file ##########
printRes = setNames(data.frame(matrix(ncol = 3, nrow = 1)), c("rules", "runtime", "accurac
printRes$rules = Num_rules
printRes$runtime = round(runtime$toc - runtime$tic,2)
printRes$accuracy =  round(accuracy,digits =3)
printRes$precision =  round(Weighted_precision,digits =3)
printRes$recall =  round(Weighted_recall,digits =3)
printRes$F1 = round(Weighted_F1,digits =3)
printRes$AUC = round(AUC_result,digits =3)

inspect(sort(rules(RCAR+_model), by = "weight"))
```