Discovering Rule Lists with Preferred Variables

Ioanna Papagianni[0000−0002−7801−4088], Matthijs van Leeuwen[0000−0002−0510−3549]

LIACS, Leiden University, the Netherlands
{i.papagianni,m.van.leeuwen}@liacs.leidenuniv.nl

Abstract. Interpretable machine learning focuses on learning models that are inherently understandable by humans. Even such interpretable models, however, must be trustworthy for domain experts to adopt them. This requires not only accurate predictions, but also reliable explanations that do not contradict a domain expert's knowledge. When considering rule-based models, for example, rules may include certain variables either due to artefacts in the data, or due to the search heuristics used. When such rules are provided as explanations, this may lead to distrust.
We investigate whether human guidance could benefit interpretable machine learning when it comes to learning models that provide both accurate predictions and reliable explanations. The form of knowledge that we consider is that of *preferred variables*, i.e., variables that the domain expert deems important enough to be given higher priority than the other variables. We study this question for the task of multiclass classification, use probabilistic rule lists as interpretable models, and use the minimum description length (MDL) principle for model selection.
We propose S-Classy, an algorithm based on beam search that learns rule lists and takes preferred variables into account. We compare S-Classy to its baseline method, i.e., without using preferred variables, and empirically demonstrate that adding preferred variables does not harm predictive performance, while it does result in the preferred variables being used in rules higher up in the learned rule lists.

Keywords: classification · probabilistic rule lists · minimum description length (MDL) principle · human-guided machine learning

1 Introduction

Explainable Artificial Intelligence (XAI) and interpretable machine learning [10] are important topics that currently attract a lot of attention within and outside the academic community. Although the two fields are clearly related in that both aim to provide explanations for predictions (or other outcomes) given by AI systems, they usually refer to slightly different approaches. XAI approaches typically attempt to provide post-hoc explanations for predictions [16], which can be done for any type of predictive model—whether it's a complex, 'black box' model such as a neural network, or a simpler model such as a linear regression model. Interpretable machine learning, on the other hand, focuses on learning

interpretable models, models that are inherently understandable by humans—such as linear regression models and rule- and tree-based models.

In application domains where high-stake decisions are made, such as law and health care, predictive models are used for *decision support*, i.e., assisting the domain experts making decisions rather than autonomously making decisions. This calls for *human-centred AI*, where machine learning models augment human experts rather than replace them. This leads to extra requirements on models and algorithms: for domain experts to adopt an AI system, it must be trustworthy, i.e., it must not only provide accurate predictions, but also reliable explanations.

Providing reliable explanations is by no means a simple feat. Models are often learned from relatively small datasets—especially in high-stake settings where data is typically expensive—and certain associations may be perceived as more reliable than others. In health care, for example, a medical doctor will only trust explanations using patient properties of which they think a relationship with the target variable is plausible. Explanations that contradict a domain expert's knowledge, in contrast, are likely to be detrimental to their trust.

We argue that interpretable machine learning has an advantage over XAI in such settings, because interpretable models make it easier to explain how and why predictions are made. Nevertheless, this does not imply that the predictions made by interpretable models are "right for the right reasons" [15]. When considering rule-based models, for example, rules may be based on certain variables either due to associations in parts of the data, or due to the search heuristics used. When such rules are provided as explanations, this will lead to distrust.

Approach and contributions. In this paper we investigate whether human guidance could benefit interpretable machine learning when it comes to learning models that provide both accurate predictions and reliable explanations. More specifically, we study whether prior knowledge provided by a domain expert may lead to models consistent with that knowledge. This can be seen as an instance of *informed machine learning* [17], in which prior knowledge (given as, e.g., knowledge graphs or human feedback) informs the learning process.

The form of knowledge that we consider is that of *preferred variables*, i.e., variables that the domain expert deems important enough to be given higher priority than the other variables while learning a predictive model. The idea is that specifying detailed knowledge is often hard, but experts will usually have a good idea of which variables they expect to be the most informative with regard to the variable of interest, for which predictions are to be made.

We consider the task of multiclass classification, because it is one of the most commonly studied and practically used machine learning tasks. As models we use probabilistic rule lists, because they are interpretable and we have recently introduced algorithms for finding good rule lists using the minimum description length (MDL) principle as model selection criterion [11,12]. That is, we use compression as optimisation criterion, which has as most notable advantages that it makes hyper-parameter tuning unnecessary and avoids overfitting.

After discussing related work in Section 2, we motivate and formalise the problem of discovering rule lists with preferred variables in Section 3. Following

this, in Section 4 we propose S-Classy, an algorithm based on beam search that learns rule lists and takes preferred variables into account by first exploring and considering rules that include at least one preferred variable. Section 5 empirically investigates the effect of having preferred variables on compression, runtime, predictive accuracy, overfitting, and rule list size. For this we simulate external knowledge by ranking variables by feature importances that we obtained with random forests. We compare S-Classy to its baseline method bClassy, which does not use the preferred variables, on commonly used benchmark datasets. The results demonstrate that augmenting the rule learning process with background knowledge—in the form of preferred variables—does not harm any of the major evaluation criteria, while it does result in the preferred variables being used in rules higher up in the rule list.

2 Related Work

Based on the type of model used, rule learning can be roughly divided in *rule list* learning, *rule set* learning, and mixtures of both. Well-known classification algorithms such as Ripper [3], C4.5 [13], Furia [6], and unordered-CN2 [2] use an ordered *one-vs-all* approach to learn rules for the multiclass classification problem, as a result of which they essentially return ordered lists of rule sets. Such lists of sets are harder to interpret than 'plain' rule lists or rule sets. CBA [8] uses large numbers of association rules, which also hampers interpretability.

Direct rule set learning methods include IDS [7] and DRS [19], but unlike our approach they are not probabilistic. Turs [18] is a recent method for learning 'truly unordered' probabilistic rule sets, using a surrogate score to tackle incomplete rule sets. Another recent approach is Classy [12], a state-of-the-art algorithm for learning ordered rule lists. It discovers probabilistic rules for multinomial targets with both categorical and quantitative predictive variables. Both Turs and Classy use the minimum description length (MDL) principle [5] as model selection criterion to select rules that compress the data well but have a relatively low model complexity. While Classy uses a pre-mined set of candidate patterns, Proença et al. [11] later proposed SSD++, an improved algorithm that directly finds good rule lists using beam search for candidate generation. Although SSD++ was introduced for subgroup list discovery, it can just as well be be used for classification; we will call this beam search version bClassy.

As far as we are aware, how to influence search in rule learning using background knowledge has hardly been studied. In subgroup discovery, IDSD [4] is an interactive search where the user can influence the beam of a beam search by providing feedback (like/dislike). This results in erratic search behaviour though.

3 Rule Learning with Preferred Variables

We start with important definitions and notation in Subsection 3.1, after which we introduce the problem statement in Subsection 3.2 and briefly summarise the model and data encoding that we will use in Subsection 3.3.

3.1 Data, Rules, and Rule Lists

Let $D = (X, Y)$ be a supervised dataset, consisting of a dataset X and a (multi)class label vector Y. Let $\mathcal{X}$ and $\mathcal{Y}$ be the instance space and the set of all $|\mathcal{Y}|$ classes, respectively. Let $V = \{v_1, v_2, ..., v_m\}$ be the set of all $m = |V|$ variables in X, with each v_i representing a one-dimensional variable with domain $\text{dom}(v_i)$. Each $(x, y) \in D$ is a record, where instance $x = (\mathtt{x_1}, \mathtt{x_2}, ..., \mathtt{x_m}) \in \mathcal{X}$ is a vector of values with $\mathtt{x_i} \in \text{dom}(v_i)$ for each v_i, and $y \in \mathcal{Y}$ is the class label belonging to the instance. Dataset D has $n = |D|$ records.

We are interested in learning rules from data. Here, a *rule* r is a conditional statement that links occurrences of patterns to class probabilities. More precisely, a rule is a pair $r = (p, \pi(p))$, where antecedent p is a pattern and its consequent is a probability distribution $\pi(p)$. A *pattern* is a conjunction of conditions over variables, e.g., $p = [v_1 = \text{'A'} \wedge v_3 = 0]$. Further, $\pi(p)$ is a categorical probability distribution $\pi = (\pi^{y_1}, \pi^{y_2}, ..., \pi^{y_{|\mathcal{Y}|}})$ over all class labels $\mathcal{Y}$. An example rule could be *if* $[v_1 = \text{'A'} \wedge v_3 = 0]$ *then* $\pi^{y_1} = 0.85, \pi^{y_2} = 0.05, \pi^{y_3} = 0.10$.

A *probabilistic rule list* (PRL) R is an ordered list of $l+1$ rules $(r_1, r_2, ..., r_l, r_\emptyset)$, where the last rule in the list, $r_\emptyset$, is called the default rule. It has the empty set as antecedent and is assigned a probability distribution $\pi_\emptyset$.

The *usage* of a pattern $p \in R$ is the number of its occurrences in a dataset D, disregarding all instances that were covered by previous patterns in R, i.e.,

$$usg(p_i \,|\, R, D) = |\{x \subset D \,|\, p_i \sqsubseteq x \wedge (\bigwedge_{\forall j<i} p_j \not\sqsubseteq x)\}|, \tag{1}$$

where $p \sqsubseteq x$ denotes that pattern p *occurs* in instance x, i.e., x satisfies all conditions in p, and $\not\sqsubseteq$ is the reverse. The *label-oriented usage* of a pattern $p_i \in R$ is the number of pattern occurrences in dataset D that correspond to class label l, where $D^{y=l} = \{(x, y) \subset D \,|\, y = l\}$ is the subset of D where class label l occurs:

$$usg(p_i \,|\, R, D^{y=l}) = |\{x \subset D^{y=l} \,|\, p_i \sqsubseteq x \wedge (\bigwedge_{\forall j<i} p_j \not\sqsubseteq x)\}|. \tag{2}$$

We consider the problem of rule learning for *multiclass (or multinomial) classification*, meaning that it is our aim to learn a rule list from a given supervised dataset such that it can accurately predict the class labels for unseen instances.

3.2 Problem Statement

As mentioned in the previous section, the minimum description length (MDL) principle [5] has previously been successfully used for rule learning [12,18]. Informally, the principle states that the best model is the one that best compresses the data together with the model. Formally, given a (training) supervised dataset D and a corresponding model class $\mathcal{R}$, consisting of all possible rule lists for D, the optimal rule list R^* is given by

$$R^* = \underset{R \in \mathcal{R}}{\text{argmin}}\, L(D, R) = \underset{R \in \mathcal{R}}{\text{argmin}}\, [L(R) + L(Y|X, R)]\,, \tag{3}$$

where $L(R)$ is the encoded length, in bits, of the rule list and $L(Y|X, R)$ is the encoded length, in bits, of class labels Y given data X and rule list R.

This is the same problem formalisation as was previously used for CLASSY [12], and was shown to result in compact rule lists that performed well in terms of predictive performance. One advantage of using the MDL principle is that it automatically protects from overfitting by balancing model complexity with goodness of fit, hence cross-validation for hyperparameter tuning is not necessary.

Since finding the optimal rule list R^* is a hard problem, heuristic algorithms—such as CLASSY and BCLASSY—are used in practice. Although predictive performance of the resulting rule lists may be excellent, the patterns used may be less than ideal to a domain expert due to two reasons: 1) the optimal rule list may not be found due to the use of heuristic search; and 2) under certain circumstances multiple variables may lead to equally 'good' rules, in which case one of those is arbitrarily chosen and used. The latter may happen, for example, when two variables are strongly associated. For high-stake decisions it is crucial that a model uses the 'right' variables for a prediction though, so that the patterns can be served to domain experts as explanations and gain their trust.

Because of the second reason, improving the learning algorithms is unlikely to ever completely address this issue: in practice only a limited sample of data is available, and that may contain insufficient information to be able to choose the 'right' variables. We therefore argue that it may be needed to integrate *external knowledge* in the learning process in order to obtain more reliable explanations.

As an initial step in this direction, we investigate whether injecting limited expert knowledge may be helpful in guiding the heuristic search to rule lists that are at least equally predictive but use patterns that are potentially more informative to domain experts than if no such knowledge is provided.

More specifically, we assume that we have access to a *domain expert* who is knowledgeable on the domain of the classification problem under consideration. The domain expert specifies a (small) set of *preferred variables* $U \subset V$ of which they are convinced they could and should be used for predicting the target variable Y. The preferred variables should be given higher priority during the search for a rule list than the remaining variables, i.e., $V \setminus U$, meaning that they should be considered for pattern growth first. Note that this does not mean that the preferred variables should be used regardless of the data; if the domain expert is wrong, this should not result in models with poor predictive performance.

3.3 Encoding

For the code length of the model and the code length of the data given the model, i.e., $L(R)$ and $(Y|X, R)$, respectively, we use the same encoding as used by CLASSY [12]. We here only provide a brief overview.

Model encoding. We use the *universal code for integers*[1] $L_{\mathbb{N}}(i)$ to penalise for rule length, while the *uniform code* provides a means to assign equal-length codes to variables and values of variables. The length of a pattern p_i is given

[1] $L_{\mathbb{N}}(i) = \log^* i + \log \lambda$, where $\log^* i = \log i + \log\log i + ...$ and constant $\lambda \approx 2.865064$.

by the number of conditions in that pattern encoded by the universal code for integers, and then each condition c_j in p_i is encoded using uniform codes for the variables and values, i.e., $L(p_i) = L_{\mathbb{N}}(|p_i|) + \sum_{c_j \in p_i}(\log |V| + \log \text{dom}(v_j))$. Now, the total length of a probabilistic rule list R is computed as the sum of the number of rules and the lengths of the individual patterns, given by

$$L(R) = L_{\mathbb{N}}(|R|) + \sum_{p_i \in R} L(p_i). \tag{4}$$

Data encoding. For the encoding of the class label vector the *prequential plug-in code* is used, which at each stage is the optimal code given the data so far (i.e., it *sequentially* predicts the next symbol). It is given by

$$\pi_{plug-in}(y_i = l|Y_{i-1}) := \frac{|\{y \in Y_{i-1}|y = l\}| + \epsilon}{\sum_{k \in \mathcal{Y}} |\{y \in Y_{i-1}|y = k\}| + \epsilon}, \tag{5}$$

where y_i is the i^{th} class label, $Y_{i-1} = \{y_1, ..., y_{i-1}\}$ is the sequence of the $i-1$ first class labels, and $\epsilon = 1$ (for a uniform prior). The above can be expressed by the maximum likelihood estimator (MLE) π_i^l for any probability $\pi(y = l \,|\, p_i)$, any rule p_i and any class label l. The Laplace smoothing (pseudocount ϵ) is added to the equation of the maximum likelihood estimator to all label-oriented usages to avoid probabilities of zero. Then, the *smoothed* MLE is formalised as

$$\pi_i^l = \frac{usg(p_i \,|\, R, D^{y=l}) + \epsilon}{usg(p_i \,|\, R, D) + |\mathcal{Y}|\epsilon}. \tag{6}$$

4 Beam search with preferred variables

Rule learning is generally a hard problem, and finding the MDL-optimal rule list is certainly hard—heuristic algorithms are therefore common practice. The original Classy algorithm [12] iteratively selected patterns from a pre-mined candidate set. The SSD++ algorithm [11] improved on this by means of a beam search; although SSD++ is aimed at finding subgroup sets, Proença's dissertation [9] has shown that rule learning and subgroup discovery are closely related. We here employ the SSD++ beam search algorithm for learning rule lists as in Classy, and dub this beam search variant of the algorithm bClassy. Using the beam search has several advantages: 1) there is no need to pre-mine candidates, allowing to prune the search space as the search progresses; 2) on-the-fly discretisation can be used, giving better results for quantitative variables [11]. An additional advantage that is of particular importance to us is that the beam search allows to guide the search using preferred variables.

We propose S-Classy, a greedy algorithm based on bClassy that starts with a rule list consisting of only the default rule and iteratively adds rules until compression cannot be improved, taking into account the preferred variables as given by the domain expert. Before we describe the algorithm in detail we briefly summarise how compression gain is computed.

Algorithm 1 S-Classy algorithm

Input: Dataset D, set of preferred variables S, beam width ω_b, maximum pattern length $|r|_{max}$, minimum support threshold m_s
Output: Probabilistic rule list R

1: $R \leftarrow [r_\emptyset]$ ▷ Start with default rule
2: **while** True **do** ▷ Repeat while compression can be improved
3: $Cands \leftarrow \emptyset$
4: **for** $s \in S$ **do** ▷ Perform beam search for each preferred variable
5: $Cands \leftarrow Cands \cup BeamSearch(s, R, D, \omega_b, |r|_{max}, m_s)$
6: **end for**
7: **if** $Cands = \emptyset$ **then** ▷ No candidates? Perform full beam search
8: $Cands \leftarrow BeamSearch(\emptyset, R, D, \omega_b, |r|_{max}, m_s)$
9: **end if**
10: $r \leftarrow \text{argmax}_{r' \in Cands}\, \delta L(D, R \oplus r')$ ▷ Pick rule that maximises normalised gain
11: **if** $\delta L(D, R \oplus r) > 0$ **then**
12: $R \leftarrow R \oplus r$ ▷ Add rule to rule list
13: $S \leftarrow S \setminus VariablesInPattern(r)$ ▷ Update preferred variable set
14: **else**
15: **return** R ▷ Return final rule list
16: **end if**
17: **end while**

Compression gain. To find a good rule list according to Eq 3, in each iteration we aim to find that rule that improves compression the most. To this end, absolute compression gain ΔL is defined as $\Delta L(D, R \oplus r) = L(D, R) - L(D, R \oplus r)$, i.e., the number of bits gained by adding a rule r to rule list R. Since greedy search combined with absolute compression gain has been shown to favour fewer rules that are less accurate but cover more instances [12], we also use the *normalised compression gain*. The normalised gain $\delta L(D, R \oplus r)$ is the absolute gain normalised by the usage of the corresponding pattern p, $\delta L(D, R \oplus r) = \frac{\Delta L(D, R \oplus r)}{usg(p \mid R, D)}$.

Algorithm. S-Classy is given by Algorithm 1. It starts by initialising a rule list to the default rule (Ln 1). Then, one rule is added in each iteration of the main loop (Ln 2–17) until no rule that improves compression can be found and the resulting rule list is returned (Ln 15). In each iteration, first a beam search is done for each preferred variable (Ln 4–6), starting the search from the given preferred variable s (i.e., it only considers patterns that include a condition on s). If S is empty or the previous beam searches did not result in any candidate rules, then a beam search considering all possible rules is conducted (Ln 7–9). In all calls to the beam search, the beam width, maximum pattern length, and minimum support threshold hyper-parameters are given to constrain the search.

After all beam search procedures have been completed, the candidate rule with the largest normalised compression gain is selected (Ln 10). If its compression gain is larger than 0 (Ln 11), it improves overall compression (Eq. 3) and is thus added to the rule list (Ln 12). Note that all rules are added at the end of the rule list, but just before the default rule (which is always updated to reflect the class distribution of the uncovered instances). Finally, the set of preferred

Table 1. *Dataset characteristics*: number of records ($|D|$); total number of values for all v_i, denoted $|\mathtt{x}|$; number of variables m; number of class labels ($|\mathcal{y}|$); size of set S, denoted K. Further, $*$ indicates dataset characteristics after the removal of rows with *NaN* values. *Relative compression* ($L\%$) and *runtime* (sec) are averaged over all 10 folds using the top-K set of preferred variables S.

Dataset	Characteristics					$\mu(L\%)$		$\mu(sec)$	
	$\|D\|$	$\|\mathtt{x}\|$	m	$\|\mathcal{y}\|$	K	S-Classy	bClassy	S-Classy	bClassy
Breast*	683	16	9	2	1	**0.62**	**0.62**	**0.12**	0.13
Cong. voting*	231	34	16	2	2	**0.75**	**0.75**	**0.37**	0.42
Dermatology*	358	49	12	6	2	**0.47**	**0.47**	**3.09**	3.11
Heart*	297	50	13	5	2	0.12	**0.13**	**1.14**	1.21
Ionosphere	351	157	34	2	4	**0.39**	**0.39**	**3.35**	3.48
Iris	150	19	4	3	1	**0.75**	**0.75**	**0.31**	0.33
Led7	3200	24	7	10	1	**0.51**	**0.51**	6.21	**6.11**
Letter	20000	102	16	26	2	**0.50**	**0.50**	**822.08**	823.55
Mushroom*	5644	80	21	2	3	**0.97**	**0.97**	**1.94**	1.99
Pen digits	10992	86	16	10	2	**0.84**	**0.84**	294.91	**294.09**
Pima Indians	768	38	8	2	1	**0.10**	**0.10**	**0.01**	**0.01**
Tic-tac-toe	958	29	9	2	1	**0.46**	**0.46**	**1.22**	1.26
Waveform	5000	101	21	3	3	**0.44**	**0.44**	**42.75**	42.76
Wine	178	68	13	3	3	0.62	**0.64**	**2.78**	2.94

variables is updated: any of the preferred variables used in the pattern of rule r are removed from S, and they will not be given priority in further iterations.

Note that the algorithm only adds rules that improve overall compression. In that sense the preferred variables can help guide the search, but if no viable rules using those variables are found the algorithm falls back to using other variables—if the knowledge provided by a domain expert is not in agreement with the evidence in the data, then this cannot have a negative impact.

5 Experiments

All experiments[2] use a minimum support threshold of $m_s = 5\%$ and maximum pattern length of $|r|_{max} = 4$, following the baseline comparisons of Proença [12].

Data. We evaluate our algorithm using 14 discrete-valued datasets publicly available from LUCS/KDD[3], see Table 1 for their characteristics. We randomise the order of the instances before splitting into folds for 10-fold cross-validation.

Simulating knowledge. As no expert knowledge is available for these datasets and we aim for reproducible results, we choose to simulate expert knowledge. For fairness we do not wish to use bClassy or other interpretable models for this.

[2] The source code is available at: `https://github.com/ioannapap/S-CLASSY`.

[3] `https://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html#datasets`

Instead we use Random Forest (RF), which is widely implemented and often used in real world applications [1,14]. We train a random forest on the entire dataset and rank all variables based on the entropy-based feature importance scores. Next, we select a set S of K variables as preferred variables, where K depends on the number of variables in a dataset. We use $K = m \cdot 10\%$, rounded upwards, which is a small but substantial percentage of the total number of variables m. For our empirical evaluation, we consider three sets of variables as preferred variables to be given as input: 1) top-K, the K highest ranked variables; bottom-K, the K lowest ranked variables; and 3) random-K, K variables that are selected uniformly at random from V (once, before all experiments are done).

Evaluation criteria. We evaluate the algorithm on 1) compression, 2) runtime, 3) classification performance, 4) overfitting, and 5) interpretability. For the compression criterion, we calculate *relative compression* gain as

$$L\% = 1 - \frac{L(D, R)}{L(D, \{\emptyset\})}. \tag{7}$$

That is, it is the compressed size of the data given the final rule list $L(D, R)$ over the compressed size of the data given the rule list with only the default rule $L(D, \{\emptyset\})$. We subtract the fraction from 1, so that the closer to 1 the relative compression gets, the better. We use a timer to measure runtime in seconds for every fold and then average it over all 10 folds ($\mu(sec)$). Similar to [12], we check the *Area Under the ROC Curve* (AUC) to quantify the *classification performance*. We weigh per class binary $AUCs$ with their marginal frequencies since the majority of the datasets are multinomial.

Overfitting is here evaluated as the mean absolute difference in AUC between the train and test set, i.e., $|\mu(AUC)_{train} - \mu(AUC)_{test}|$. How to evaluate the interpretability of a rule-based model is a complex topic on itself. A minimum requirement for a rule list to be interpretable is that it needs to be small, i.e., it must contain relatively few rules that are not too long. We therefore quantify *interpretability* using average rule length ($\mu|r|$) and the average number of rules in a rule list ($\mu|R|$). Lastly, we are interested in investigating whether the preferred variables influenced the rules learned. For this we use the frequencies f and positions of the preferred variables in the learned rules. Specifically, we care mostly about the frequency of preferred variables in the first rule of each rule list, which we annotate by $f@1$ and average over all folds.

5.1 Results

Regarding relative compression and runtime, we recognise that when the top-K preferred variable set S is used, S-Classy performs similarly to bClassy, see Table 1. S-Classy's runtime is slightly lower overall. Our algorithm also ranks first regarding accuracy when using the top-K, as seen in Table 2. This is a good result, as it shows that adding preferred variables does not harm predictive performance and may even benefit it. When using the bottom-K variables as background knowledge, performance is worse than for the other variants and

Table 2. Mean (μ) results per dataset using 10-fold cross-validation, with fixed $m_s = 5\%$, $\omega_b = 100$ and $|r|_{max} = 4$. Bottom, random and top are the three different sets of simulated 'preferred variables' used for S-Classy. *AUC* is the *Area Under the ROC Curve* in test set, $|\mu(AUC)_{train} - \mu(AUC)_{test}|$ is the overfitting. Then, f@1 and max (f@1) is the frequency of *all* preferred variables and the maximum (highest) frequency of a preferred variable respectively, both at 1^{st} position of the rule list, for S-Classy (abbreviated S-Cl) and bClassy (abbreviated bCl).

Dataset	$\mu(AUC)_{test}$					$\lvert\mu(AUC)_{train} - \mu(AUC)_{test}\rvert$					$\mu(f@1)$		max (f@1)	
	S-Classy			bClassy	RF	S-Classy			bClassy	RF	S-Cl	bCl	S-Cl	bCl
	bottom	random	top			bottom	random	top			top		top	
Breast	0.50	0.94	0.94	0.94	**0.95**	**0**	0.007	0.007	0.007	**0**	**1**	**1**	**1**	**1**
Cong. voting	**0.97**	**0.97**	**0.97**	**0.97**	0.96	**0.002**	**0.002**	**0.002**	**0.002**	0.007	**0.50**	**0.50**	**1**	**1**
Dermatology	0.50	**0.86**	**0.86**	**0.86**	0.73	**0**	0.029	0.034	0.029	0.006	**0.50**	0	**0.60**	0
Heart	0.50	0.66	**0.67**	0.66	**0.67**	**0**	0.018	0.018	0.019	0.012	0.95	**1**	**1**	**1**
Ionosphere	0.54	0.84	0.85	0.85	**0.90**	**0.004**	0.081	0.072	0.075	0.009	0.25	**0.47**	**1**	**1**
Iris	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.019**	**0.019**	**0.019**	**0.019**	0.022	**1**	0	**1**	0
Led7	**0.84**	**0.84**	**0.84**	**0.84**	0.50	0.005	0.005	0.005	0.005	**0.001**	**1**	**1**	**1**	**1**
Letter	**0.79**	**0.79**	**0.79**	**0.79**	0.50	0.007	0.008	0.007	0.011	**0**	**1**	0.45	**1**	0.90
Mushroom	0.50	**1**	**1**	**1**	0.99	**0**	**0**	**0**	**0**	**0**	**0.33**	**0.33**	**1**	**1**
Pen digits	0.97	0.97	**0.98**	0.97	0.50	0.010	0.010	0.011	0.011	**0**	**0.33**	**0.33**	**1**	**1**
Pima Indians	0.50	0.50	0.66	0.66	**0.67**	**0**	**0**	0.001	0.001	0.012	**1**	**1**	**1**	**1**
Tic-tac-toe	**0.87**	**0.87**	**0.87**	**0.87**	0.64	**0.009**	**0.009**	**0.009**	**0.009**	0.025	**1**	**1**	**1**	**1**
Waveform	0.82	0.82	**0.83**	0.82	0.77	0.026	0.026	0.026	0.026	**0.006**	**0.57**	0.50	**1**	**1**
Wine	0.92	0.92	0.92	0.92	**0.94**	0.058	0.058	0.058	0.058	**0.034**	**0.37**	0.33	**1**	0.90
$rank_{all}$	2.86	1.86	**1.29**	1.57	3.14	**1.36**	2.21	2.36	2.57	2.36	**1.14**	1.36	**1**	1.29

also worse than bClassy. This indicates that providing preferred variables is not entirely without risk: despite our goals, poorly chosen variables may result in worse predictive performance. All variants of S-Classy perform better overall than RF ($rank_{all}$)[4], which is interesting because the RF-based feature importance does benefit S-Classy. With regard to overfitting, bottom and random sets do well but that is to be expected; poor predictions on training data are still poor on test data. More interestingly, the results suggest that providing informative preferred variables potentially leads to less overfitting, as S-Classy with 'top' evaluation sets ranks higher than bClassy and equal to RF with regard to overfitting.

Clearly, the average number[5] of conditions and rules that were discovered in all different sets of S-Classy and bClassy, presented in Table 3, show little to no difference with bClassy scoring overall first. However, when we calculate the *Jaccard distance*[6] between the conditions of bClassy rules and S-Classy rules per rule position, we discover that the rules are in fact different. The non-zero *Jaccard distances* shown in Table 3 are further explained by Table 2, where we present the mean frequency f of all variables $s \in S$ in the top-K set and the *maximum* frequency f of the most used preferred variable at the first rule position in the rule list. Moreover, Figure 1 shows—for four datasets—in more detail the differences in the variables used in the rules learned by top-K S-Classy and bClassy, making it visible that our algorithm manages to

[4] $Rank_{all}$ (smaller is better) is the average rank over all datasets.
[5] The lowest (> 0) $\mu|r|$, $\mu|R|$ the better, 0 is treated as the worst.
[6] For *Jaccard distance*, the closer to 0 the more similar and the closer to 1 the more different (preferred state).

Table 3. Mean (μ) number of 1) conditions in a rule r ($|r|$); and 2) rules in a rule list R ($|R|$) per experiment, using 10-fold cross validation with fixed $m_s = 5\%$, $\omega_b = 100$ and $|r|_{max} = 4$. Bottom, random and top are the simulated sets of 'preferred variables' used for S-CLASSY. The mean (μ) *Jaccard distance* is calculated between each simulated variable set with bCLASSY.

Dataset	$\mu\|r\|$				$\mu\|R\|$				$\mu(Jaccard\ distance)$		
	S-CLASSY			bCLASSY	S-CLASSY			bCLASSY	S-CLASSY		
	bottom	random	top		bottom	random	top		bottom	random	top
Breast	0	**4**	**4**	**4**	0	**2**	**2**	**2**	**1**	0	0
Cong. voting	2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0.47**	0	0
Dermatology	**1**	2	2	2	**1**	7	7	7	**1**	0.07	0.34
Heart	0	3	3	**2**	0	**2**	**2**	**2**	**1**	0.37	0.37
Ionosphere	**1**	2	2	2	**1**	6	5	5	**1**	0.66	0.04
Iris	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	0.23	0.23	**0.47**
Led7	**3**	**3**	**3**	**3**	**21**	**21**	**21**	**21**	**0**	**0**	**0**
Letter	**4**	**4**	**4**	**4**	153	**150**	151	151	**0.91**	0.67	0.68
Mushroom	**1**	2	2	2	**1**	2	5	5	**1**	0.85	0
Pen digits	**4**	**4**	**4**	**4**	77	**73**	76	76	**0.94**	**0.94**	0
Pima Indians	0	0	**1**	**1**	0	0	**1**	**1**	**1**	**1**	0
Tic-tac-toe	4	**3**	**3**	**3**	**5**	**5**	**5**	**5**	0.13	**0.24**	0
Waveform	4	4	**3**	4	**39**	**39**	40	**39**	0.76	**0.81**	0.27
Wine	3	3	3	**2**	**4**	**4**	**4**	**4**	**0.08**	**0.08**	**0.08**
$rank_{all}$	2.14	1.57	1.36	**1.29**	2	1.5	1.64	**1.43**	**1.21**	1.71	2.21

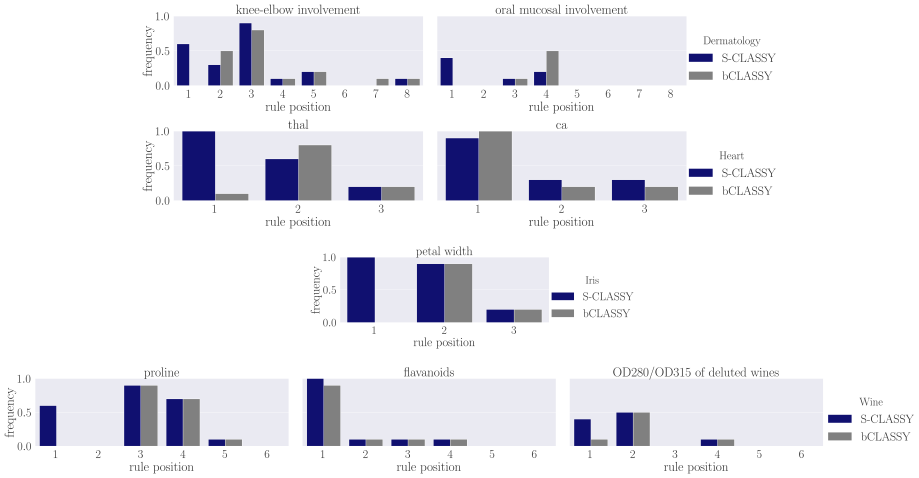

Fig. 1. Mean (μ) frequency of top preferred variables per rule position in the rule list using 10-fold cross validation in Dermatology, Heart, Iris and Wine dataset.

include and combine the preferred variables more often than bCLASSY as earlier as possible. These results not only demonstrate that our method manages to learn different rules in the rule list from its predecessor algorithm, but at the same time 1) ensure that the preferred variables are incorporated at the very beginning of our rule list, and 2) keep competitive, and even in some experiments higher, classification performance.

6 Conclusions & Future Work

We argued that human guidance might be beneficial to interpretable machine learning, especially in settings where predictions need to be accurate as well as reliable and trustworthy explanations are needed. We investigated whether this is the case for the problem of multiclass classification and used rule lists as models. The form of knowledge that we considered is that of *preferred variables*, i.e., variables that the domain expert deems important enough to be given higher priority in the learning process than the other variables.

We proposed S-Classy, an algorithm based on beam search that learns rule lists and takes preferred variables into account by first only exploring rules that include one of the preferred variables. An empirical comparison of S-Classy to its baseline method, i.e., without using preferred variables, demonstrated that adding preferred variables does not harm predictive performance, while it does result in the preferred variables being used in rules higher up in the learned rule lists. From this we conclude that human guidance might indeed be beneficial to rule learning, for predictive accuracy but also for learning the 'right' rules.

We consider this only to be a first step towards human-guided rule learning. In the future, interesting directions would be to examine other model classes, such as (unordered) rule sets, and expand the background knowledge language, e.g., by allowing constraints based on conditions or patterns, or based on other properties of a rule-based model. A more extensive study on the consequences of using specified/preferred variables in terms of classification performance and interpretability is also worth pursuing. In addition, we aim to evaluate our approach with real-world case studies involving actual domain knowledge provided by domain experts. Finally, we think that interactive rule learning is a promising avenue for future research.

Acknowledgements. This work is supported by Project 4 of the Digital Twin research programme, a TTW Perspectief programme with project number P18-03 that is primarily financed by the Dutch Research Council (NWO).

References

1. Chaudhary, A., Kolhe, S., Kamal, R.: An improved random forest classifier for multi-class classification. Information Processing in Agriculture **3**(4), 215–222 (2016)
2. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. In: European Working Session on Learning. pp. 151–163 (1991)
3. Cohen, W.W.: Fast effective rule induction. In: Machine learning (1995)
4. Dzyuba, V., Leeuwen, M.v.: Interactive discovery of interesting subgroup sets. In: IDA. pp. 150–161 (2013)
5. Grünwald, P.D.: The Minimum Description Length Principle (Adaptive Computation and Machine Learning) (2007)
6. Hühn, J., Hüllermeier, E.: FURIA: an algorithm for unordered fuzzy rule induction. Data Mining and Knowledge Discovery **19**(3), 293–319 (2009)

7. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: A joint framework for description and prediction. In: ACM SIGKDD. pp. 1675–1684 (2016)
8. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In: ACM SIGKDD (1998)
9. Manuel Proenca, H.: Robust rules for prediction and description. Ph.D. thesis, Leiden University (2021)
10. Molnar, C.: Interpretable machine learning. Lulu.com (2020)
11. Proença, H.M., Grünwald, P., Bäck, T., van Leeuwen, M.: Robust subgroup discovery. Data Mining and Knowledge Discovery **36**(5), 1885–1970 (2022)
12. Proença, H.M., van Leeuwen, M.: Interpretable multiclass classification by MDL-based rule lists. Information Sciences **512**, 1372–1393 (2020)
13. Quinlan, J.: C4.5: programs for machine learning (2014)
14. Sarica, A., Cerasa, A., Quattrone, A.: Random forest algorithm for the classification of neuroimaging data in Alzheimer's disease: a systematic review. Frontiers in aging neuroscience **9**, 329 (2017)
15. Schramowski, P., Stammer, W., Teso, S., Brugger, A., Herbert, F., Shao, X., Luigs, H.G., Mahlein, A.K., Kersting, K.: Making deep neural networks right for the right scientific reasons by interacting with their explanations. Nature Machine Intelligence **2**(8), 476–486 (2020)
16. Sokol, K., Flach, P.: Explainability is in the mind of the beholder: Establishing the foundations of explainable artificial intelligence. arXiv preprint arXiv:2112.14466 (2021)
17. Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., et al.: Informed Machine Learning–A Taxonomy and Survey of Integrating Knowledge into Learning Systems. arXiv:1903.12394 (2019)
18. Yang, L., van Leeuwen, M.: Truly Unordered Probabilistic Rule Sets for Multi-class Classification. In: ECMLPKDD (2022)
19. Zhang, G., Gionis, A.: Diverse rule sets. In: ACM SIGKDD. pp. 1532–1541 (2020)