A Link between Pattern Structures and Random Forests

Tilo Krause[1], Lars Lumpe[2] and Stefan E. Schmidt[3]

Institut für Algebra,
Technische Universität Dresden
tilokrause@hotmail.de[1], larslumpe@gmail.com[2], midt1@msn.com[3]

Abstract. Our paper shows that decision trees and random forests can be described via pattern structures. This new perspective leads to a better understanding of random forests and delivers a new way of analyzing complex data with the help of pattern structures.

1 Introduction

Pattern structures within the framework of formal concept analysis have been introduced in [5]. Since then, they have been the subject of further investigation like in [3, 11, 13, 12, 14] and have turned out to be an important tool for analyzing various real-world applications (cf. [5, 7–10]). Previous investigations like [6] show the connection between decision trees and formal concept analysis. In this paper we want to present a new application and a link between pattern structures and random forests. But, first, we will introduce embedded pattern structures, enabling us to create pattern structures from pattern setups. Subsequently, we are going to apply our findings to a data set [4] and train a random forest to predict the quality of red wines. Our approach leads to a better understanding of pattern structures and random forests. Also we provide a useful visualization to interpret a random forest and construct a model to predict classes of red wines.

2 Preliminaries

We will start with a few general definitions:

Definition 1 (Adjunction)**.** Let $\mathbb{P} = (P, \leq)$ and $\mathbb{L} = (L, \leq)$ be posets; furthermore let $f : P \to L$ and $g : L \to P$ be maps.

(1) The pair (f, g) is an **adjunction** w.r.t. $(\mathbb{P}, \mathbb{L})$ if $fx \leq y$ is equivalent to $x \leq gy$ for all $x \in P$ and $y \in L$. In this case, we will refer to $(\mathbb{P}, \mathbb{L}, f, g)$ as a **poset adjunction**.

(2) g is **residual** from $\mathbb{L}$ to $\mathbb{P}$ if the preimage of a principal filter in $\mathbb{P}$ under g is always a principal filter in $\mathbb{L}$, that is, for every $x \in P$ there exists $y \in L$ s.t.

$$g^{-1}\{s \in P \mid x \leq s\} = \{t \in L \mid y \leq t\}.$$

Definition 2 (restriction)**.** Let $\mathbb{P} := (P, \leq_{\mathbb{P}})$ be a poset; then, for every set U the poset

$$\mathbb{P} \mid U := (U, \leq_{\mathbb{P}} \cap (U \times U))$$

is called the **restriction** of $\mathbb{P}$ onto U.

Definition 3. Let $f : A \to B$ be a surjective map, then

$$IP_f : B \to 2^A,\ b \mapsto f^{-1}b$$

is a partition of A. We call IP_f an f **induced partition of A**.

Note: We write $f : A \twoheadrightarrow B$ for a surjective map f from A to B.

If we consider a poset of patterns, it often arises as a dual of a given poset.

Definition 4 (Dedekind MacNeille completion)**.** Let $\mathbb{D} := (D, \leq)$ be an ordered set. For every subset $A \subseteq D$ we call

$$A^{\uparrow} := \{d \in D \mid a \leq d \text{ for all } a \in A\}$$

the set of all upper bounds of A. The set of all lower bounds of A is defined dually and denoted by $A_{\downarrow}$. A **cut** of $\mathbb{D}$ is a pair (A, B) with $A, B \subseteq D$, $A^{\uparrow} = B$ and $A = B_{\downarrow}$. It is well known that these cuts ordered

$$(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2\ (\Leftrightarrow B_2 \subseteq B_1)$$

form a complete lattice, the **Dedekind MacNeille completion**. It is the smallest complete lattice containing a subset that is order isomorphic with $\mathbb{D}$ and denoted by $DMN(\mathbb{D})$.

Definition 5 (Interval Poset)**.** Let $\mathbb{P} := (P, \leq_{\mathbb{P}})$ be a poset. Then we call

$$\text{Int}\mathbb{P} := \{[p, q]_{\mathbb{P}} \mid p, q \in P\} \text{ with } [p, q]_{\mathbb{P}} := \{t \in P \mid p \leq_{\mathbb{P}} t \leq_{\mathbb{P}} q\}$$

the **set of all intervals** of $\mathbb{P}$, and we refer to $\mathbb{I}\text{nt}\mathbb{P} := (\text{Int}\mathbb{P}, \supseteq)$ as the interval poset of $\mathbb{P}$.

Remark: (i) Let $\mathbb{P}$ be a poset. Then $\mathbb{I}\text{nt}\mathbb{P}$ is an upper bounded poset, that is, $\emptyset$ is the greatest element of $\mathbb{I}\text{nt}\mathbb{P}$, provided $\mathbb{P}$ has at least 2 elements. Furthermore, if $\mathbb{P}$ is a (complete) lattice, then so is $\mathbb{I}\text{nt}\mathbb{P}$.
(ii) If A is a set of attributes, then $(\mathbb{R}^A, \leq) := (\mathbb{R}, \leq)^A$ is a lattice. With (i) it follows that $\mathbb{I}\text{nt}(\mathbb{R}^A, \leq)$ is a lower bounded lattice.

The main definitions of FCA are based on a binary relation I between a set of so called objects G and a set of so called attributes M. However, in many real-world knowledge discovery problems, researchers have to deal with data sets that are much more complex than binary data tables. In our case, there was a set of numerical attributes, such as the amount of acetic acid, density, the amount of salt, etc., describing the quality of a red wine. To deal with this kind of data, pattern structures are a useful tool.

Definition 6 (pattern setup, pattern structure)**.** A triple $\mathcal{P} = (G, \mathbb{D}, \delta)$ is a **pattern setup** if G is a set, $\mathbb{D} = (D, \sqsubseteq)$ is a poset, and $\delta : G \to D$ is a map. In case every subset of $\delta G := \{\delta g \mid g \in G\}$ has an infimum in $\mathbb{D}$, we will refer to $\mathcal{P}$ as **pattern structure**.

In the sequel we need definition 5 and an application of Theorem 1 from [13]. We listed both below.

Definition 7 (pattern morphism)**.** If $\mathcal{G} = (G, \mathbb{D}, \delta)$ and $\mathcal{H} = (H, \mathbb{E}, \varepsilon)$ each is a pattern setup, then a pair (f, φ) forms a **pattern morphism** from $\mathcal{G}$ to $\mathcal{H}$ if $f : G \to H$ is a map and φ is a residual map from $\mathbb{D}$ to $\mathbb{E}$ satisfying $\varphi \circ \delta = \varepsilon \circ f$, that is, the following diagram is commutative:

$$\begin{tikzcd}
G \arrow[r, "f"] \arrow[d, "\delta"'] & H \arrow[d, "\varepsilon"] \\
\mathbb{D} \arrow[r, "\varphi"'] & \mathbb{E}
\end{tikzcd}$$

Theorem 1. *Let $\mathcal{G}$ be a pattern structure and $\mathcal{H}$ be a pattern setup. If (f, φ) is a pattern morphism from $\mathcal{G}$ to $\mathcal{H}$, with f being surjective, then $\mathcal{H}$ is also a pattern structure.*

In this paper we want to present a connection between pattern structures and random forests. Decision trees are an important data mining tool, used to predict classes of data sets. Many applications of decision trees build a model on a so called training set and then try to predict classes of unseen data (test sets). For an introduction into the subject, we recommend [1]. Before we give an abstract definition of a decision tree, we want to present a way to construct one.

Construction 1 (decision tree)**.** We start at the root node and split the data on the feature that results in the best splitting measure (e.g. least gini impurity). In an iterative process, we repeat this splitting procedure at each child node until the nodes are pure (contain only one class) or a termination rule is fulfilled. Nodes that do not split the data any further are called leaves.

Definition 8 (decision tree)**.** A **finite tree** is defined as a triple $\mathbb{T} := (T, \leq_{\mathbb{T}}, 0_{\mathbb{T}})$ consisting of a finite poset $(T, \leq)$ with least element $0_{\mathbb{T}}$. This least element will be denoted as **root node**. Every principal downset is partially ordered. The maximal elements of $\mathbb{T}$ will be called **leaves** of $\mathbb{T}$; let $Le\mathbb{T}$ denote the set of leaves of $\mathbb{T}$. A triple $\mathcal{T} := (G, \mathbb{T}, \lambda)$ will be called a **finite decision tree** or short **decision tree** if G is a finite set, $\mathbb{T} := (T, \leq_{\mathbb{T}}, 0_{\mathbb{T}})$ is a finite tree and $\lambda : G \twoheadrightarrow LeT$ is a surjective map. The map

$$\tau : T \to LeT, t \mapsto \{b \in LeT \mid t \leq b\}$$

allocates the set of leaves to a node, where the node is less than or equal to the leaf (in the sense of $\leq_{\mathbb{T}}$). We call

$$\mathbb{T}^c := (T^c, \leq_{\mathbb{T}}, 0_{\mathbb{T}}) \text{ with } T^c := T \cup \{1_{\mathbb{T}}\}$$

the **tree completion** of $\mathbb{T}$ and $\mathbb{T}^c$ a **complete tree**. This leads to a map

$$\overline{\lambda} : G \to T^c,\ g \mapsto \lambda g.$$

In the here presented context we will refer to $\mathcal{T}^s := (G, \mathbb{T}, \mathbb{T}^c, \lambda, \overline{\lambda}, \tau)$ as a **decision tree setup**.

Remark: (i) A tree completion $\mathbb{T}^c$ is the smallest complete lattice which contains $\mathbb{T}$.
(ii) The λ induced partition of G is given by

$$IP_{\lambda} : LeT \to 2^G,\ b \mapsto \lambda^{-1} b$$

The following example illustrates the definition of a decision tree.

G	m_1	m_2	class
w_0	4	14	0
w_1	5	12	0
w_2	6	11	1
w_3	7	10	1
w_4	8	13	0
w_5	9	15	0

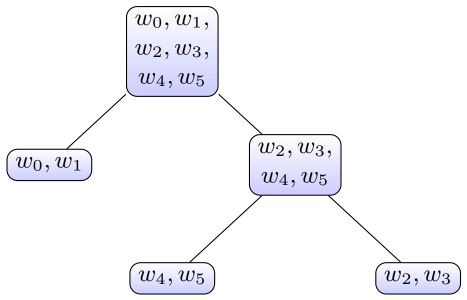

Fig. 1. Decision tree example. left: the data; right: the constructed tree

In [2] Leo Breiman introduced random forests, which lead to significant improvements in classification accuracy by growing an ensemble of decision trees and letting them vote for the most popular class. In this paper we use the following definition of a random forest.

Definition 9 (random forest)**.** Let $\mathcal{T}_i^s := (G, \mathbb{T}_i, \mathbb{T}_i^c, \lambda_i, \overline{\lambda_i}, \tau_i)$ be a decision tree setup for all $i \in I$, then we call

$$\mathbb{F} := (G, \mathbb{T}_{prod}, \lambda) \text{ with}$$

$$T_{prod} := \prod_{i \in I} T_i,\ \mathbb{T}_{prod} := \prod_{i \in I} \mathbb{T}_i \text{ and } \lambda : G \to \prod_{i \in I} T_i, g \mapsto \{\lambda_i g\}_{i \in I}$$

a **random forest**.

3 Embedded Pattern Structures

The following definition establishes the connection between pattern setups and pattern structures.

Definition 10 (embedded pattern structure)**.** Let $\mathbb{L}$ be a complete lattice and $\mathcal{P} := (G, \mathbb{D}, \delta)$ a pattern setup with $\mathbb{D} := \mathbb{L}|D$. Then we call

$$\mathcal{P}_e := (G, \mathbb{L}, \mathbb{D}, \overline{\delta}) \text{ with } \overline{\delta} : G \to L, g \mapsto \delta g$$

the **embedded pattern structure**. We say the pattern setup $\mathcal{P}$ is **embedded** in the pattern structure $(G, \mathbb{L}, \overline{\delta})$.

The following two constructions are a demonstration of how to build an embedded pattern structure from a pattern setup.

Construction 2. Let G be a set and let $\mathbb{D} := (D, \leq)$ be a poset, then for every map $\varrho : G \to D$ the **elementary pattern structure** is given by:

$$\mathcal{P}_\varrho := (G, \mathbb{L}, \varepsilon) \text{ with } \mathbb{L} := (2^D, \supseteq) \text{ and } \varepsilon : G \to 2^D, g \mapsto \{\varrho g\}.$$

Hence, the pattern setup $(G, \mathbb{D}, \varrho)$ is embedded in the pattern structure $\mathcal{P}_\varrho$.

Construction 3. For every pattern setup $\mathcal{P} := (G, \mathbb{D}, \delta)$, a pattern structure is given through the Dedekind MacNeille completion and

$$(G, DMN(\mathbb{D}), \overline{\delta}) \text{ with } \overline{\delta} : G \to DMN(\mathbb{D}), g \mapsto \delta g$$

4 Making the Link

In this section we are going to present a way to construct a pattern structure from a given decision tree. Starting point is the following pattern setup:

Construction 4. Let $\mathcal{T}^s := (G, \mathbb{T}, \mathbb{T}^c, \lambda, \overline{\lambda}, \tau)$ be a decision tree setup, then

$$\mathcal{P} := (G, \mathbb{T}, \lambda)$$

is a pattern setup. We embed this pattern setup in the embedded pattern structure

$$\mathcal{P}^e := (G, \mathbb{T}^c, \mathbb{T}, \overline{\lambda}).$$

This leads to the pattern structure $(G, \mathbb{T}^c, \overline{\lambda})$.

This construction provides a connection between decision trees and pattern structures. Below we show the link between pattern structures and random forests. For this purpose we have to look at the product of pattern structures.

Construction 5. Let I be a finite set and $\mathcal{P}_i := (G, \mathbb{D}_i, \delta_i)$ a pattern structure for every $i \in I$. Then

$$\mathcal{P} := (G, \mathbb{D}, \delta)$$

with

$$D := \prod_{i \in I} D_i,\ \mathbb{D} := \prod_{i \in I} \mathbb{D}_i \text{ and } \delta : G \to \prod_{i \in I} D_i,\ g \mapsto \{\delta_i g\}_{i \in I}$$

is a pattern structure, too.

Our construction shows that a product of pattern structures leads to a pattern structure again. A decision tree can be described via a pattern structure and, since a random forest is a product of decision trees, a random forest is characterized by a pattern structure, too. The following theorem lays this out.

Construction 6. Let I be an index set and $\mathcal{T}_i^s := (G, \mathbb{T}_i, \mathbb{T}_i^c, \lambda_i, \overline{\lambda_i}, \tau_i)$ a decision tree setup for all $i \in I$, and $\mathbb{F} := (G, \mathbb{T}_{prod}, \lambda)$ the corresponding random forest. Construction 4 provides us the embedded pattern structures $\mathcal{P}_i^e := (G, \mathbb{T}_i^c, \mathbb{T}_i, \overline{\lambda}_i)$ for every tree $\mathbb{T}_i$. Then

$$\mathcal{P} := (G, \mathbb{T}, \lambda) \text{ with}$$

$$T := \prod_{i \in I} T_i^c,\ \mathbb{T} := \prod_{i \in I} \mathbb{T}_i^c \text{ and } \lambda : G \to \prod_{i \in I} T_i^c,\ g \mapsto \{\overline{\lambda_i} g\}_{i \in I}$$

is a pattern structure, too.

Construction 6 presents a way to describe a random forest through a pattern structure. Not only does this novel perspective enable a deeper understanding of random forests, but it also proves useful in real world applications, as will be pointed out in the next section.

5 Real World Application

In this section, we are going to first describe the public data set we used, and then apply Construction 6 to a typical data-mining related classification problem.

5.1 Red Wine Data Set

To apply our previous results on a public data set we choose the red wine data set from [4]. There are 1599 examples of wines, described by 11 numerical attributes. The input include objective tests (e.g. fixed acidity, sulphates, PH values, residual sugar, chlorides, density, alcohol...) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality between 0 (very bad) and 10 (very excellent). For our purpose, we established a binary distinction where every wine with a quality score above 5 is classified "good" and all below as "bad". This led to a set of 855 positive and 744 negative examples. We split the data into a training set (75% of the examples) and a test set (25% of the examples) and trained a random forest with onehundred trees with at least 10 examples per leaf and a maximal depth of 3 on the training set. We verified our model on the test set and obtained the following confusion matrix:

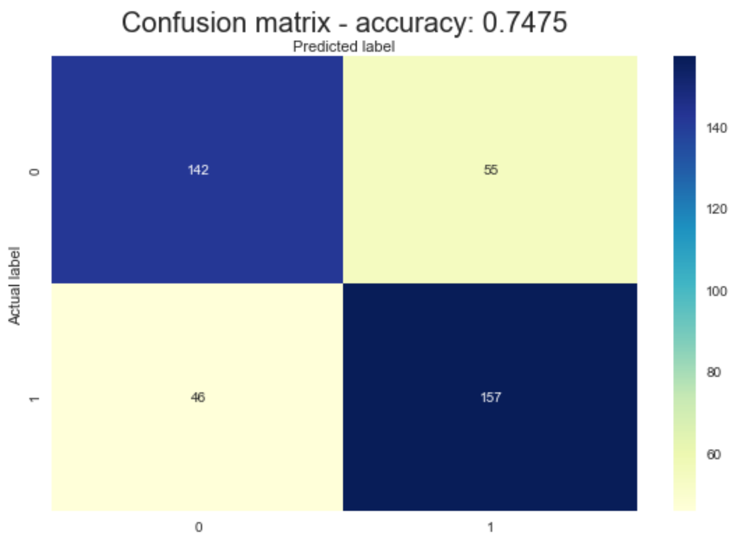

Fig. 2. model of the random forest on the test set

Random forests are an useful tool like this example shows, but it is hard to comprehend how a decision tree gives a ruling. Interval pattern structures, introduced in [8], can be helpful like the next section shows.

5.2 From Evaluation Map to Interval Pattern Structures

Many data mining relevant data sets (like the red wine data set) can be described by an evaluation matrix:

Definition 11 (evaluation map)**.** Let G be a finite set, M a set of attributes, and $\mathbb{W}_m := (W_m, \leq_m)$ a complete lattice for every attribute $m \in M$. Further, let

$$W := \prod_{m\in M} W_m \text{ and } \mathbb{W} := \prod_{m\in M} \mathbb{W}_m.$$

Then, a map

$$\alpha : G \to W, g \mapsto \prod_{m\in M} \{\alpha_m g\}$$

such that

$$\alpha_m : G \to W_m, g \mapsto w_m$$

is called **evaluation map**. We call $\mathcal{E} := (G, M, \mathbb{W}, \alpha)$ an **evaluation setup**.

Example 1. *In the wine data set in [4] it is possible to interpret the wines as a set G, the describing attributes as the set M, and $\mathbb{W}_m$ as the numerical range of attribute m with the natural order.*

In the above example, the evaluation map $\alpha : G \to W$ assigns to every wine a vector with values of all attributes $m \in M$.

Construction 7. Let $\mathcal{E} := (G, M, \mathbb{W}, \alpha)$ be an evaluation setup and, further, let $\mathcal{T}^s := (G, \mathbb{T}, \mathbb{T}^c, \lambda, \overline{\lambda}, \tau)$ be a decision tree setup. Then,

$$\varphi : T^c \to \mathrm{Int}\mathbb{W}|\varphi T^c,\ t \mapsto [\inf_{\mathbb{W}} \alpha(\lambda^{-1}\tau t),\ \sup_{\mathbb{W}} \alpha(\lambda^{-1}\tau t)]$$

maps intervals to all nodes of the decision tree. The intervals are spanned by the objects in the node.

It is possible to create a pattern structure from this map, as the following theorem shows.

Theorem 2. *Let $\mathcal{E} := (G, M, \mathbb{W}, \alpha)$ be an evaluation setup and $\mathcal{T}^s := (G, \mathbb{T}, \mathbb{T}^c, \lambda, \overline{\lambda}, \tau)$ a decision tree setup. Then*

$$\mathcal{P} := (G, \mathrm{Int}\mathbb{W}|\varphi T^c, \varphi \circ \overline{\lambda}) \text{ with}$$
$$\varphi : T^c \to \mathrm{Int}\mathbb{W}|\varphi T^c,\ t \mapsto [\inf_{\mathbb{W}} \alpha(\lambda^{-1}\tau t),\ \sup_{\mathbb{W}} \alpha(\lambda^{-1}\tau t)]$$

is a pattern structure.

Proof. We want to apply theorem 1. Since the identity map is clearly surjective, we have to show that (id, φ) is a pattern morphism, more precisely we have to prove that φ is a residual map. This can be achieved by applying definition 1. The preimage of a principal filter in $\mathrm{Int}\mathbb{W}|\varphi T$ under φ is always a principal filter in $\mathbb{T}^c$. For every $x \in \mathrm{Int}\mathbb{W}|\varphi T$

$$\varphi^{-1}\{s \in \mathrm{Int}\mathbb{W}|\varphi T \mid x \leq s\} = \{t \in T^c \mid \varphi^{-1}x \leq t\}$$

holds. □

To lift the previous theorem up to a random forest, we use construction 5.

Construction 8. Let $\mathcal{E} := (G, M, \mathbb{W}, \alpha)$ be an evaluation setup, I an index set, and $\mathbb{F} := \prod_{i \in I} \mathbb{T}_i$ a random forest with corresponding decision tree setups $\mathcal{T}_i^s := (G, \mathbb{T}_i, \mathbb{T}_i^c, \lambda_i, \overline{\lambda_i}, \tau_i)$. Then

$$\mathcal{P}_i := (G, \mathrm{Int}\mathbb{W}|\varphi_i T_i^c, \varphi_i \circ \overline{\lambda_i}) \text{ with}$$

$$\varphi_i : T_i^c \to \mathrm{Int}\mathbb{W}|\varphi_i T_i^c,\ t_i \mapsto [\inf_{\mathbb{W}} \alpha(\lambda_i^{-1}\tau_i t_i),\ \sup_{\mathbb{W}} \alpha(\lambda_i^{-1}\tau_i t_i)]$$

is a pattern structure and

$$\mathcal{P} := (G, \mathbb{D}, \delta)$$

with

$$D := \prod_{i \in I} \mathrm{Int}\mathbb{W}|\varphi_i T_i^c, \mathbb{D} := \prod_{i \in I} \mathbb{I}\mathrm{nt}\mathbb{W}|\varphi_i T_i^c \text{ and } \delta : G \to \prod_{i \in I} \mathrm{Int}\mathbb{W}|\varphi_i T_i^c,\ g \mapsto \{\varphi_i \circ \overline{\lambda_i} g\}_{i \in I}$$

is a pattern structure, too.

This construction provides a set of intervals for every $g \in G$. The intersection of these intervals is not empty because g is at least in every interval. Every g assigned a predicted class probability from a decision tree, which is a purity measure. It is calculated as the ratio of good examples to all examples in a leaf. For a random forest, the average of this ratio for all leaves, which contains g are used to calculate a score. We took the best scored g of our training set and looked at union of the intervals constructed by the random forest. We received the following result:

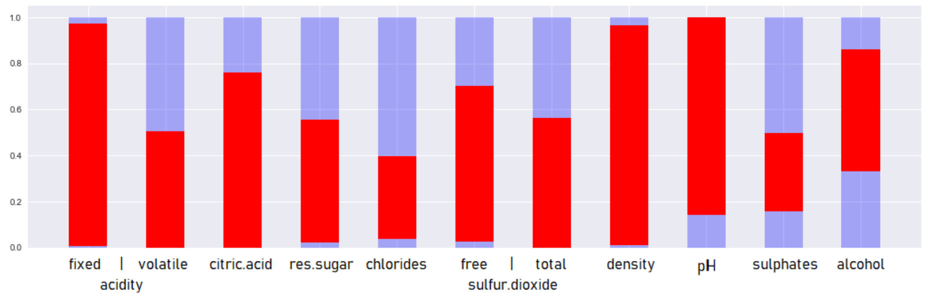

Fig. 3. Interval 1: build from the best scored wine (features scaled in the range $[0, 1]$)

The diagram show all 11 attributes (fixed acidity, volatile acidity,...) of the wines. Displayed is the interval, which is constructed from best scored wine of the random forest. This pattern contains 330 wines of the training set, 297 of them are good and 33 are bad. As mentioned before the training set contains of 75% of the data, which are 1199 wines. 652 of them are good and 547 bad.
To choose a collection of intervals for a prediction model we calculate a score on a pattern p via

$$score(p) = \frac{good\ wines\ in\ p}{all\ good\ wines} + ([ratio_p * 100] - 50)/50.$$

To improve interpretability, we selected only the following best scored 5 intervals and add them to the first interval to build our model. Again we looked at the union of the intervals and scaled the range of the features to $[0, 1]$.

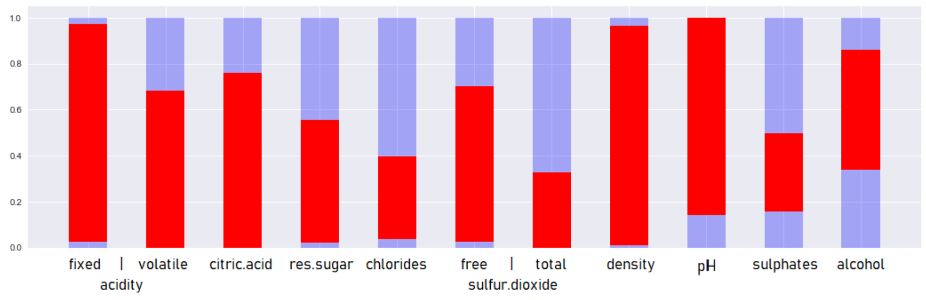

Fig. 4. Interval 2: contains altogether 331 wines, 295 good and 36 bad

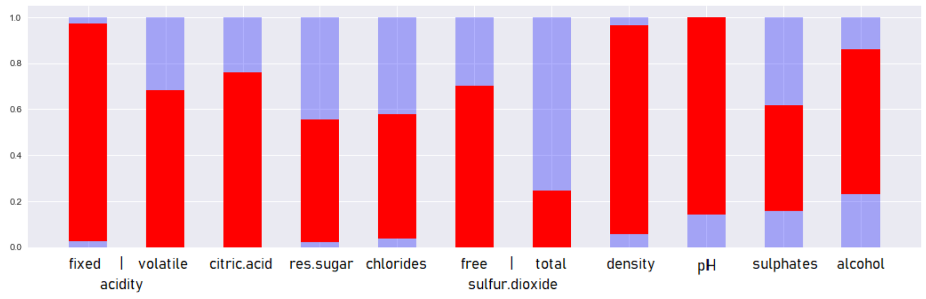

Fig. 5. Interval 3: contains altogether 466 wines, 383 good and 83 bad

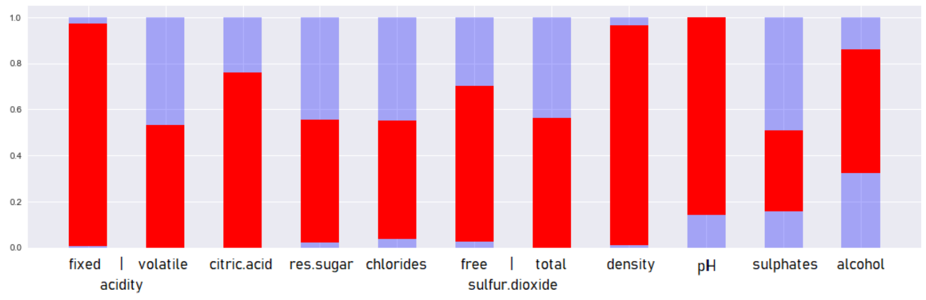

Fig. 6. Interval 4: contains altogether 366 wines, 318 good and 48 bad

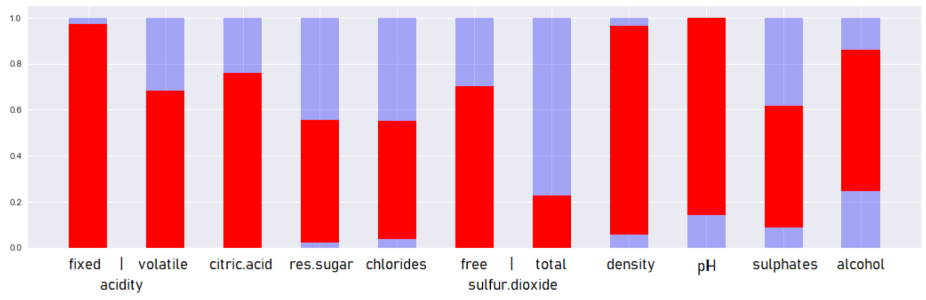

Fig. 7. Interval 5: contains altogether 466 wines, 452 good and 142 bad

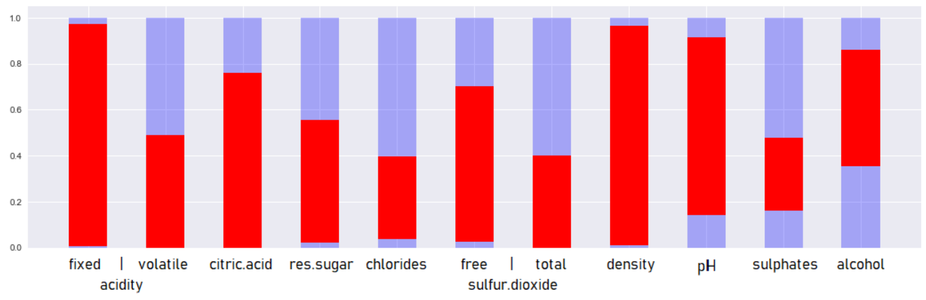

Fig. 8. Interval 6: contains altogether 288 wines, 261 good and 27 bad

Combining these 6 intervals to predict the classes of the test set leads to the following confusion matrix:

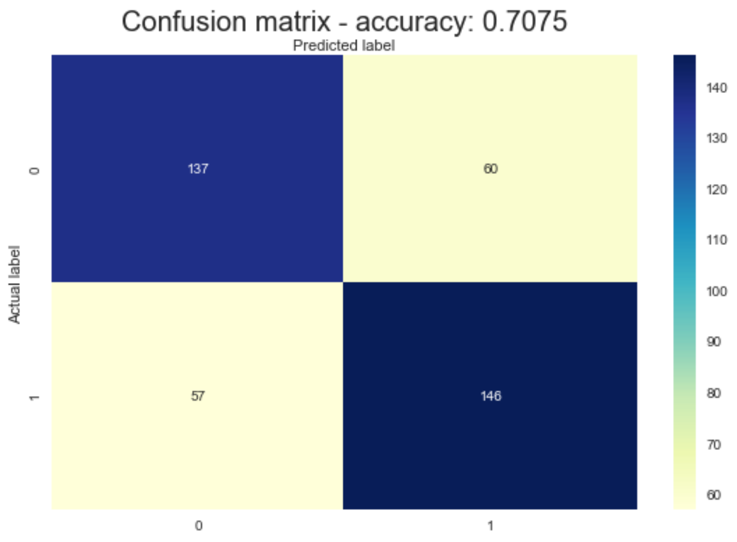

Fig. 9. Combination of the 6 intervals

Our model lost some prediction quality, but the high interpretability makes our algorithm useful. For example fixed acidity, which is the first attribute, seems not important for the quality of a wine since all patterns have a huge range in this feature. On the other hand the range in the attribute sulphates is comparatively small. This suggests the assumption, that sulphates are an important feature to classify a wine.

6 Conclusion

We introduced a novel framework for the application of pattern structures. The paper presents a new way of building a pattern structure through a given decision tree. This link is an interesting view on things and via a visualization it leads to a better understanding of how a decision tree gives a ruling. Furthermore, our paper shows that the product of pattern structures are also pattern structures and so we extend the link between pattern structures and decision trees to random forests, since they are a product of decision trees. Also we introduced a model to predict classes of red wines. This model is an abstraction of a random forest. As here introduced the prediction of the random forest is better, but the high interpretability makes our algorithm valuable. This is just a first useful example of our method. Further investigations on other data sets have to prove the importance of our algorithm.
Besides, we made some theoretical progress in the field of pattern structures by introducing embedded pattern structures. They provide a framework where it is possible to construct a pattern structure out of a given pattern setup.

References

1. L. Breiman, J. H. Friedman, R. A. Olshen, C.J. Stone (1984), Classification and Regression Trees, Chapman & Hall.
2. L. Breiman (2001), Machine Learning, Springer.
3. A. Buzmakov, S. O. Kuznetsov, A. Napoli (2015) , Revisiting Pattern Structure Projections. Formal Concept Analysis. Lecture Notes in Artificial Intelligence (Springer), Vol. 9113, pp 200-215.
4. P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis (2009), Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems 47(4), pp. 547-553.
5. B. Ganter, S. O. Kuznetsov (2001), Pattern Structures and Their Projections. Proc. 9th Int. Conf. on Conceptual Structures, ICCS'01, G. Stumme and H. Delugach (Eds.). Lecture Notes in Artificial Intelligence (Springer), Vol. 2120, pp. 129-142.
6. S. Guillas, K. Bertet, J.M. Ogier, N. Girard (2008), Some links between decision tree and dichotomic lattice, CLA 2008, pp. 193-205
7. T. B. Kaiser, S. E. Schmidt (2011), Some remarks on the relation between annotated ordered sets and pattern structures. Pattern Recognition and Machine Intelligence. Lecture Notes in Computer Science (Springer), Vol. 6744, pp 43-48.
8. M. Kaytoue, S. O. Kuznetsov, A. Napoli, S. Duplessis (2011), Mining gene expression data with pattern structures in formal concept analysis. Information Sciences (Elsevier), Vol.181, pp. 1989-2001.
9. S. O. Kuznetsov (2009), Pattern structures for analyzing complex data. In H. Sakai et al. (Eds.). Proceedings of the 12th international conference on rough sets, fuzzy sets, data mining and granular computing (RSFDGrC'09). Lecture Notes in Artificial Intelligence (Springer), Vol. 5908, pp. 33-44.
10. S. O. Kuznetsov (2013), Scalable Knowledge Discovery in Complex Data with Pattern Structures. In: P. Maji, A. Ghosh, M.N. Murty, K. Ghosh, S.K. Pal, (Eds.). Proc. 5th International Conference Pattern Recognition and Machine Intelligence

(PReMI'2013). Lecture Notes in Computer Science (Springer), Vol. 8251, pp. 30-41.

11. L. Lumpe and S. E. Schmidt (2015), A Note on Pattern Structures and Their Projections. Formal Concept Analysis. Lecture Notes in Artificial Intelligence (Springer), Vol. 9113, pp 145-150.
12. L. Lumpe, S. E. Schmidt (2016), Morphisms Between Pattern Structures and Their Impact on Concept Lattices, FCA4AI@ ECAI 2016, pp 25-34.
13. L. Lumpe, S. E. Schmidt (2015), Pattern Structures and Their Morphisms. CLA 2015, pp. 171-179.
14. L. Lumpe, S. E. Schmidt (2016), Viewing Morphisms Between Pattern Structures via Their Concept Lattices and via Their Representations, International Symposium on Methodologies for Intelligent Systems 2017, pp. 597-608.