Journal of Machine Learning Research x (xxxx) x-x Submitted 6/09; Revised 9/09; Published xx/xx

An Efficient Explanation of Individual Classifications using Game Theory

Erik Štrumbelj ERIK.STRUMBELJ@FRI.UNI-LJ.SI
Igor Kononenko IGOR.KONONENKO@FRI.UNI-LJ.SI
Faculty of Computer and Information Science
University of Ljubljana
Tržaska 25, 1000 Ljubljana, Slovenia

Editor:

Abstract

We present a general method for explaining individual predictions of classification models. The method is based on fundamental concepts from coalitional game theory and predictions are explained with contributions of individual feature values. We overcome the method's initial exponential time complexity with a sampling-based approximation. In the experimental part of the paper we use the developed method on models generated by several well-known machine learning algorithms on both synthetic and real-world data sets. The results demonstrate that the method is efficient and that the explanations are intuitive and useful.

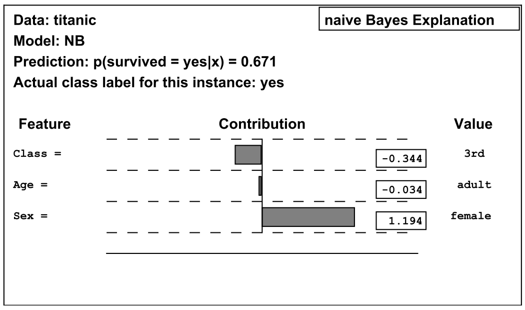
Keywords: Data Postprocessing, Classification, Explanation, Visualization.

1. Introduction

Acquisition of knowledge from data is the quintessential task of machine learning. The data are often noisy, inconsistent, and incomplete, so various preprocessing methods are used before the appropriate machine learning algorithm is applied. The knowledge we extract this way might not be suitable for immediate use and one or more *data postprocessing* methods could be applied as well. Data postprocessing includes the integration, filtering, evaluation, and *explanation* of acquired knowledge. The latter is the topic of this paper.

To introduce the reader with some of the concepts used in this paper, we start with a simple illustrative example of an explanation for a model's prediction (see Fig. 1). We use Naive Bayes because its prediction can, due to the assumption of conditional independence, easily be transformed into *contributions of individual feature values* - a vector of numbers, one for each feature value, which indicates how much each feature value contributed to the Naive Bayes model's prediction. This can be done by simply applying the logarithm to the model's equation (see, for example, Kononenko and Kukar, 2007; Becker et al., 1997).

In our example, the contributions of the three feature values can be interpreted as follows. The prior probability of a Titanic passenger's survival is 32% and the model predicts a 67% chance of survival. The fact that this passenger was female is the sole and largest contributor to the increased chance of survival. Being a passenger from the third class and an adult both speak against survival, the latter only slightly. The actual class label for this instance is "yes", so the classification is correct. This is a trivial example, but providing the end-user with such an explanation on top of a prediction, makes the prediction easier to understand and to trust. The latter is crucial in situations where important and sensitive decisions are made. One such example is medicine, where medical practitioners are known to be reluctant to use machine learning models, despite their often superior performance (Kononenko, 2001). The inherent ability of explaining its decisions is one of the main reasons for the frequent use of the Naive Bayes classifier in medical diagnosis and prognosis (Kononenko, 2001). The approach used to explain the decision in Fig. 1 is specific to Naive Bayes, but *can we design an explanation method which works for any type of classifier?* In this paper we address this question and propose a method for explaining the predictions of classification models, which can be applied to any classifier in a *uniform way.*



Data: titanic
Model: NB
Prediction: p(survived = yes|x) = 0.671
Actual class label for this instance: yes

Feature	Contribution	Value
Class =	-0.344	3rd
Age =	-0.034	adult
Sex =	1.194	female

Figure 1: An instance from the well-known Titanic data set with the Naive Bayes model's prediction and an explanation in the form of contributions of individual feature values. A copy of the Titanic dataset can be found at http://www.ailab.si/orange/datasets.psp.

1.1 Related Work

Before addressing *general* explanation methods, we list a few *model-specific* methods to emphasize two things. First, most models have model-specific explanation methods. And second, providing an explanation in the form of contributions of feature values is a common approach. Note that many more model-specific explanation methods exist and this is far from being a complete reference. Similar to Naive Bayes, other machine learning models also have an inherent explanation. For example, a decision tree's prediction is made by following a decision rule from the root to the leaf, which contains the instance. Decision rules and Bayesian networks are also examples of transparent classifiers. Nomograms are a way of visualizing contributions of feature values and were applied to Naive Bayes (Možina et al., 2004) and, in a limited way (linear kernel functions), to SVM (Jakulin et al., 2005). Other related work focusses on explaining the SVM model, most recently in the form of visualization (Poulet, 2004; Hamel, 2006) and rule-extraction (Martens et al., 2007). The ExplainD framework (Szafron et al., 2006) provides explanations for additive classifiers in the form of contributions. Breiman provided additional tools for showing how individual features contribute to the predictions of his Random Forests (Breiman, 2001). The explanation and interpretation of artificial neural networks, which are arguably one of the least transparent models, has also received a lot of attention, especially in the form of rule extraction (Towell and Shavlik, 1993; Andrews et al., 1995; Nayak, 2009).

So, *why do we even need a general explanation method?* It is not difficult to think of a reasonable scenario where a general explanation method would be useful. For example, imagine a user using a classifier and a corresponding explanation method. At some point the model might be replaced with a better performing model of a different type, which usually means that the explanation method also has to be modified or replaced. The user then has to invest time and effort into adapting to the new explanation method. This can be avoided by using a general explanation method. Overall, a good general explanation method reduces the dependence between the user-end and the underlying machine learning methods, which makes work with machine learning models more user-friendly. This is especially desirable in commercial applications and applications of machine learning in fields outside of machine learning, such as medicine, marketing, etc. An effective and efficient general explanation method would also be a useful tool for comparing how a model predicts different instances and how different models predict the same instance.

As far as the authors are aware, there exist two other general explanation methods for explaining a model's prediction: the work by Robnik-Šikonja and Kononenko (2008) and the work by Lemaire et al. (2008). While there are several differences between the two methods, both explain a prediction with contributions of feature values and both use the same basic approach. A feature value's contribution is defined as the difference between the model's initial prediction and its average prediction across perturbations of the corresponding feature. In other words, we look at how the prediction would change if we "ignore" the feature value. This myopic approach can lead to serious errors if the feature values are conditionally dependent, which is especially evident when a disjunctive concept (or any other form of redundancy) is present. We can use a simple example to illustrate how these methods work. Imagine we ask someone who is knowledgeable in boolean logic *What will the result of (*1 *OR* 1*) be?*. It will be one, of course. Now we mask the first value and ask again *What will the result of (something OR* 1*) be?*. It will still be one. So, it does not matter if the person knows or does not know the first value - the result does not change. Hence, we conclude that the first value is irrelevant for that persons decision regarding whether the result will be 1 or 0. Symmetrically, we can conclude that the second value is also irrelevant for the persons decision making process. Therefore, both values are irrelevant. This is, of course, an incorrect explanation of how these two values contribute to the persons decision.

Further details and examples of where existing methods would fail can be found in our previous work (Štrumbelj et al., 2009), where we suggest observing the changes across all possible subsets of features values. While this effectively deals with the shortcomings of previous methods, it suffers from an exponential time complexity.

To summarize, we have existing general explanation methods, which sacrifice a part of their effectiveness for efficiency, and we know that generating effective contributions requires observing the power set of all features, which is far from efficient. The contribution of this paper and its improvement over our previous work is twofold. First, we provide a rigorous theoretical analysis of our explanation method and link it with known concepts from game theory, thus formalizing some of its desirable properties. And second, we propose an efficient sampling-based approximation method, which overcomes the exponential time complexity and does not require retraining the classifier.

The remainder of this paper is organized as follows. Section 2 introduces some basic concepts from classification and coalitional game theory. In Section 3 we provide the theoretical foundations, the approximation method, and a simple illustrative example. Section 4 covers the experimental part of our work. With Section 5 we conclude the paper and provide ideas for future work.

2. Preliminaries

First, we introduce some basic concepts from classification and coalitional game theory, which are used in the formal description of our explanation method.

2.1 Classification.

In machine learning classification is a form of supervised learning where the objective is to predict the *class label* for unlabelled input instances, each described by feature values from a *feature space*. Predictions are based on background knowledge and knowledge extracted (that is, learned) from a sample of labelled instances (usually in the form of a training set).

Definition 1 *The feature space $\mathcal{A}$ is the cartesian product of n features (represented with the set $N = \{1, 2, ..., n\}$): $\mathcal{A} = A_1 \times A_2 \times ... \times A_n$, where each feature A_i is a finite set of feature values.*

Remark 1 *With this definition of a feature space we limit ourselves to finite (that is, discrete) features. However, we later show that this restriction does not apply to the approximation method, which can handle both discrete and continuous features.*

To formally describe situations where feature values are ignored, we define a subspace $\mathcal{A}_S = A'_1 \times A'_2 \times ... \times A'_n$, where $A'_i = A_i$ if $i \in S$ and $A'_i = \{\epsilon\}$ otherwise. Therefore, given a set $S \subset N$, $\mathcal{A}_S$ is a feature subspace, where features not in S are "ignored" ($\mathcal{A}_N = \mathcal{A}$). Instances from a subspace have one or more components unknown as indicated by ϵ. Now we define a classifier.

Definition 2 *A classifier, f, is a mapping from a feature space to a normalized $|C|$-dimensional space $f : \mathcal{A} \rightarrow [0, 1]^{|C|}$, where C is a finite set of labels.*

Remark 2 *We use a more general definition of a classifier to include classifiers which assign a rank or score to each class label. However, in practice, we mostly deal with two special cases: classifiers in the traditional sense (for each vector, one of the components is 1 and the rest are 0) and probabilistic classifiers (for each vector, the vector components always add up to 1 and are therefore a probability distribution over the class label state space).*

2.2 Coalitional game theory.

The following concepts from coalitional game theory are used in the formalization of our method, starting with the definition of a coalitional game.

Definition 3 *A coalitional form game is a tuple $\langle N, v \rangle$, where $N = \{1, 2, ..., n\}$ is a finite set of n players, and $v : 2^N \rightarrow \Re$ is a characteristic function such that $v(\varnothing) = 0$.*

Subsets of N are *coalitions* and N is referred to as the *grand coalition* of all players. Function v describes the *worth* of each coalition. We usually assume that the grand coalition forms and the goal is to split its worth $v(N)$ among the players in a "fair" way. Therefore, the *value* (that is, solution) is an operator ϕ which assigns to $\langle N, v \rangle$ a vector of payoffs $\phi(v) = (\phi_1, ..., \phi_n) \in \Re^n$. For each game with at least one player there are infinitely many solutions, some of which are more "fair" than others. The following four statements are attempts at axiomatizing the notion of "fairness" of a solution ϕ and are key for the axiomatic characterization of the Shapley value.

Axiom 1 $\sum\limits_{i \in N} \phi_i(v) = v(N)$. *(efficiency axiom)*

Axiom 2 *If for two players i and j $v(S \cup \{i\}) = v(S \cup \{j\})$ holds for every S, where $S \subset N$ and $i, j \notin S$, then $\phi_i(v) = \phi_j(v))$. (symmetry axiom)*

Axiom 3 *If $v(S \cup \{i\}) = v(S)$ holds for every S, where $S \subset N$ and $i \notin S$, then $\phi_i(v) = 0$. (dummy axiom)*

Axiom 4 *For any pair of games v, w : $\phi(v + w) = \phi(v) + \phi(w)$, where $(v + w)(S) = v(S) + w(S)$ for all S. (additivity axiom)*

Theorem 1 *For the game $\langle N, v \rangle$ there exists a unique solution ϕ, which satisfies axioms 1 to 4 and it is the Shapley value:*

$$Sh_i(v) = \sum_{S \subseteq N \backslash \{i\}, s=|S|} \frac{(n-s-1)!s!}{n!} (v(S \cup \{i\}) - v(S)), \qquad i = 1, ..., n.$$

Proof For a detailed proof of this theorem refer to Shapley's paper (1953). ■

The Shapley value has a wide range of applicability as illustrated in a recent survey paper by Moretti and Patrone (2008), which is dedicated entirely to this unique solution concept and its applications. From the few applications of the Shapley value in machine learning, we would like to bring to the readers attention the work of Kienan et al. (2004), who apply the Shapley value to function localization in biological and artificial networks. Their MSA framework is later used and adapted into a method for feature selection (Cohen et al., 2007).

3. Explaining Individual Predictions

In this section we provide the theoretical background. We start with a description of the intuition behind the method and then link it with coalitional game theory.

3.1 Definition of the Explanation Method

Let $N = \{1, 2, ..., n\}$ be a set representing n features, f a classifier, and $x = (x_1, x_2, ..., x_n) \in \mathcal{A}$ an instance from the feature space. First, we choose a class label. We usually chose the predicted class label, but we may choose any other class label that is of interest to us and explain the prediction from that perspective (for example, in the introductory example in Fig. 1 we could have chosen "survival = no" instead). Let c be the chosen class label and let $f_c(x)$ be the prediction component which corresponds to c. Our goal is to explain how the given feature values contribute to the *prediction difference* between the classifiers prediction for this instance and the expected prediction if no feature values are given (that is, if all feature values are "ignored"). The prediction difference can be generalized to an arbitrary subset of features $S \subseteq N$.

Definition 4 *The prediction difference $\Delta(S)$ when only values of features represented in S are known, is defined as follows:*

$$\Delta(S) = \frac{1}{|\mathcal{A}_{N \setminus S}|} \sum_{y \in \mathcal{A}_{N \setminus S}} f_c(\tau(x, y, S)) - \frac{1}{|\mathcal{A}_N|} \sum_{y \in \mathcal{A}_N} f_c(y) \tag{1}$$

$$\tau(x, y, S) = (z_1, z_2, ..., z_n), \qquad z_i = \begin{cases} x_i & ; \quad i \in S \\ y_i & ; \quad i \notin S \end{cases}$$

Remark 3 *In our previous work (Štrumbelj et al., 2009) we used a different definition: $\Delta(S) = f_c^*(S) - f_c^*(\varnothing)$, where $f^*(W)$ is obtained by retraining the classifier only on features in W and re-classifying the instance (similar to the wrappers approach in feature selection (Kohavi and John, 1997)). With this retraining approach we avoid the combinatorial explosion of going through all possible perturbations of "ignored" feature's values, but introduce other issues. However, the exponential complexity of going through all subsets of N still remains.*

The expression $\Delta(S)$ is the difference between the expected prediction when we know only those values of x, whose features are in S, and the expected prediction when no feature values are known. Note that we assume uniform distribution of feature values. Therefore, we make no assumption about the prior relevance of individual feature values. In other words, we are equally interested in each feature value's contribution. The main shortcoming of existing general explanation methods is that they do not take into account all the potential dependencies and interactions between feature values. To avoid this issue, we implicitly define interactions by defining that each prediction difference $\Delta(S)$ is composed of 2^N contributions of interactions (that is, each subset of feature values might contribute something):

$$\Delta(S) = \sum_{W \subseteq S} \mathcal{I}(W), \qquad S \subseteq N \tag{2}$$

Assuming $\mathcal{I}(\varnothing) = 0$ (that is, an interaction of nothing always contributes 0) yields a recursive definition. Therefore, function $\mathcal{I}$, which describes the interactions, always exists and is uniquely defined for a given N and function Δ:

$$\mathcal{I}(S) = \Delta(S) - \sum_{W \subset S} \mathcal{I}(W), \qquad S \subseteq N \tag{3}$$

Now we distribute the interaction contributions among the n feature values. For each interaction the involved feature values can be treated as equally responsible for the interaction as the interaction would otherwise not exist. Therefore, we define a feature value's contribution φ_i by assigning it an equal share of each interaction it is involved in.

$$\varphi_i(\Delta) = \sum_{W \subseteq N \backslash \{i\}} \frac{\mathcal{I}(W \cup \{i\})}{|W \cup \{i\}|}, \qquad i = 1, 2, ..., n \tag{4}$$

It is not surprising that we manage in some way to uniquely quantify all the possible interactions, because we explore the entire power set of the involved feature values. So, two questions arise: *Can we make this approach computationally feasible?* and *What are the advantages of this approach over other possible divisions of contributions?*. We now address both of these issues, starting with the latter.

Theorem 2 $\langle N = \{1, 2, ..., n\}, \Delta\rangle$ *is a coalitional form game and* $\varphi(\Delta) = (\varphi_1, \varphi_2, ..., \varphi_n)$ *corresponds to the game's Shapley value* $Sh(\Delta)$*.*

Proof Following the definition of Δ we get that $\Delta(\varnothing) = 0$, so the explanation of the classifier's prediction can be treated as a coalitional form game $\langle N, \Delta\rangle$. Now we provide an elementary proof that the contributions of individual feature values correspond to the Shapley value for the game $\langle N, \Delta\rangle$. The recursive definition of $\mathcal{I}$ given in Eq. (3) can be transformed into its non-recursive form:

$$\mathcal{I}(S) = \sum_{W \subseteq S} ((-1)^{|S|-|W|} \Delta(W)) \tag{5}$$

Eq. (5) can be proven by induction. We combine Eq. (4) and Eq. (5) into the following non-recursive formulation of the contribution of a feature value:

$$\varphi_i(\Delta) = \sum_{W \subseteq N \backslash \{i\}} \frac{\sum\limits_{Q \subseteq (W \cup \{i\})} ((-1)^{|W \cup \{i\}| - |Q|} \Delta(Q))}{|W \cup \{i\}|} \tag{6}$$

Let us examine the number of times $\Delta(S \cup \{i\})$, $S \subseteq N, i \notin S$, appears on the right-hand side of Eq. (6). Let $M_{\Delta(S \cup \{i\})}$ be the number of all such appearances and $k = n - |S| - 1$. The term $\Delta(S \cup \{i\})$ appears when $S \subseteq W$ and only once for each such W. For W, where $S \subseteq W$ and $|W| = |S| + a$, $\Delta(S \cup \{i\})$ appears with an alternating sign, depending on the parity of a, and there are exactly $\binom{k}{a}$ such W in the sum in Eq. (6), because we can use any combination of a additional elements from the remaining k elements that are not already in the set S. If we write all such terms up to $W = N$ and take into account that each interaction $\mathcal{I}(W)$ is divided by $|W|$, we get the following series:

$$M_{\Delta(S \cup \{i\})} = \frac{\binom{k}{0}}{n-k} - \frac{\binom{k}{1}}{n-k+1} + ... + (-1)^k \frac{\binom{k}{k}}{n} = V(n, k)$$

The treatment is similar for $\Delta(S)$, $i \notin S$ where we get $M_{\Delta(S)} = -V(n,k)$. The series $V(n,k)$ can be expressed with the beta function:

$$
\begin{aligned}
(1-x)^k &= \binom{k}{0} - \binom{k}{1}x + \binom{k}{2}x^2 - ... \pm \binom{k}{k}x^k \\
\int_0^1 x^{n-k-1}(1-x)^k\,dx &= \int_0^1 (\binom{k}{0}x^{n-k-1} - \binom{k}{1}x^{n-k} + ... \pm \binom{k}{n-1}x^{n-1})\,dx \\
B(n-k,k+1) &= \frac{\binom{k}{0}}{n-k} - \frac{\binom{k}{1}}{n-k+1} + ... \pm \frac{\binom{k}{k}}{n} = V(n,k)
\end{aligned}
$$

Using $B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}$, we get $V(n,k) = \frac{(n-k-1)!k!}{n!}$. Therefore:

$$
\begin{aligned}
\varphi_i(\Delta) &= \sum_{S \subseteq N\backslash\{i\}} V(n,n-s-1)\cdot\Delta(S\cup\{i\}) - \sum_{S \subseteq N\backslash\{i\}} V(n,n-s-1)\cdot\Delta(S) = \\
&= \sum_{S \subseteq N\backslash\{i\}} \frac{(n-s-1)!s!}{n!}\cdot(\Delta(S\cup\{i\}) - \Delta(S))
\end{aligned}
$$

■

So, the explanation method can be interpreted as follows. The instance's feature values form a coalition which causes a change in the classifier's prediction. We divide this change amongst the feature values in a way that is fair to their contributions across all possible subcoalitions. Now that we have established that the contributions correspond to the Shapley value, we take another look at its axiomatization. Axioms 1 to 3 and their interpretation in the context of our explanation method are of particular interest. The 1st axiom corresponds to our decomposition in Eq. (2) - the sum of all n contributions in an instance's explanation is equal to the difference in prediction $\Delta(N)$. Therefore, the contributions are implicitly normalized, which makes them easier to compare across different instances and different models. According to the 2nd axiom, if two features values have an identical influence on the prediction they are assigned contributions of equal size. The 3rd axiom says that if a feature has no influence on the prediction it is assigned a contribution of 0. When viewed together, these properties ensure that any effect the features might have on the classifiers output will be reflected in the generated contributions, which effectively deals with the issues of previous general explanation methods.

3.1.1 An Illustrative Example

In the introduction we used a simple boolean logic example to illustrate the shortcomings of existing general explanation methods. We concluded that in the expression (1 OR 1) both values are irrelevant and contribute nothing to the result being 1. This error results from not observing all the possible subsets of features. With the same example we illustrate how our explanation method works. We write $N = \{1,2\}$, $\mathcal{A} = \{0,1\} \times \{0,1\}$, and $x = (1,1)$. In other words, we are explaining the classifier's prediction for the expression (1 OR 1). Following the steps described in Section 3, we use Eq. (1) to calculate the $\Delta-$terms.

Intuitively, $\Delta(S)$ is the difference between the classifiers expected prediction if only values of features in S are known and the expected prediction if no values are known. If the value of at least one of the two features is known, we can predict, with certainty, that the result is 1. If both values are unknown (that is, masked) one can predict that the probability of the result being 1 is $\frac{3}{4}$. Therefore, $\Delta(1) = \Delta(2) = \Delta(1,2) = 1 - \frac{3}{4} = \frac{1}{4}$ and $\Delta(\varnothing) = \frac{3}{4} - \frac{3}{4} = 0$. Now we can calculate the interactions. $\mathcal{I}(1) = \Delta(1) = \frac{1}{4}$ and $\mathcal{I}(2) = \Delta(2) = \frac{1}{4}$. When observed together, the two features contribute less than their individual contributions would suggest, which results in a negative interaction: $\mathcal{I}(1,2) = \Delta(1,2) - (\mathcal{I}(1) + \mathcal{I}(2)) = -\frac{1}{4}$. Finally, we divide the interactions to get the final contributions: $\varphi_1 = \mathcal{I}(1) + \frac{\mathcal{I}(1,2)}{2} = \frac{1}{8}$ and $\varphi_2 = \mathcal{I}(2) + \frac{\mathcal{I}(1,2)}{2} = \frac{1}{8}$. The generated contributions reveal that both features contribute the same amount towards the prediction being 1 and the contributions sum up to the initial difference between the prediction for this instance and the prior belief.

3.2 An Approximation

We have shown that the generated contributions, φ_i, are effective in relating how individual feature values influence the classifier's prediction. Now we provide an efficient approximation. The approximation method is based on a well known alternative formulation of the Shapley value. Let $\pi(N)$ be the set of all ordered permutations of N. Let $Pre^i(\mathcal{O})$ be the set of players which are predecessors of player i in the order $\mathcal{O} \in \pi(N)$. A feature value's contribution can now be expressed as:

$$\varphi_i(\Delta) = \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} \left(\Delta(Pre^i(\mathcal{O}) \cup \{i\}) - \Delta(Pre^i(\mathcal{O})) \right), \qquad i = 1, ..., n. \tag{7}$$

Eq. (7) is the a well-known alternative formulation of the Shapley value. An algorithm for the computation of the Shapley value, which is based on Eq. (7), was presented by Castro et al. (2008). However, in our case, the exponential time complexity is still hidden in our definition of Δ (see Eq. (1)). If we use the alternative definition used in our previous work (see Remark 3), we can compute function $\Delta(S)$, for a given S, in polynomial time (assuming that the learning algorithm has a polynomial time complexity). However, this requires retraining the classifier for each $S \subseteq N$, so the method would no longer be independent of the learning algorithm and we would also require the training set that the original classifier was trained on. To avoid this and still achieve an efficient explanation method, we extend the sampling algorithm in the following way. We use a different, but equivalent formulation of Eq. (1). While the sum in this equation redundantly counts each $f(\tau(x,y,S))$ term $|\mathcal{A}_S|$ times (instead of just once) it is equivalent to Eq. (1) and simplifies the sampling procedure:

$$\Delta(S) = \frac{1}{|\mathcal{A}|} \sum_{y \in \mathcal{A}} \left(f(\tau(x,y,S)) - f(y) \right) \tag{8}$$

We replace occurrences of Δ in Eq. (7) with Eq. (8):

$$\varphi_i(\Delta) = \frac{1}{n! \cdot |\mathcal{A}|} \sum_{\mathcal{O} \in \pi(N)} \sum_{y \in \mathcal{A}} \left(f(\tau(x,y,Pre^i(\mathcal{O}) \cup \{i\})) - f(\tau(x,y,Pre^i(\mathcal{O}))) \right) \tag{9}$$

We use the following sampling procedure. Our sampling population is $\pi(N) \times \mathcal{A}$ and each order/instance pair defines one sample $X_{\mathcal{O},y\in\mathcal{A}} = f(\tau(x,y,Pre^i(\mathcal{O}) \cup \{i\})) - f(\tau(x,y,Pre^i(\mathcal{O})))$. If some features are continuous, we have an infinite population, but the properties of the sampling procedure do not change. If we draw a sample completely at random then all samples have an equal probability of being drawn ($\frac{1}{n!\cdot|\mathcal{A}|}$) and $E[X_{\mathcal{O},y\in\mathcal{A}}] = \varphi_i$. Now consider the case where m such samples are drawn (with replacement) and observe the random variable $\hat{\varphi}_i = \frac{1}{m}\sum_{j=1}^{m} X_j$, where X_j is the $j-$th sample. According to the central limit theorem, $\hat{\varphi}_i$ is approximately normally distributed with mean φ_i and variance $\frac{\sigma_i^2}{m}$, where σ_i^2 is the population variance for the $i-$th feature. Therefore, $\hat{\varphi}_i$ is an unbiased and consistent estimator of φ_i. The computation is summarized in Algorithm 1.

Algorithm 1 Approximating the contribution of the i-th feature's value, φ_i, for instance $x \in \mathcal{A}$.

```
determine $m$, the desired number of samples
$\varphi_i \leftarrow 0$
for $i = 1$ to $m$ do
    choose a random permutation of features $\mathcal{O} \in \pi(N)$
    choose a random instance $y \in \mathcal{A}$
    $v_1 \leftarrow f(\tau(x,y,Pre^i(\mathcal{O}) \cup \{i\}))$
    $v_2 \leftarrow f(\tau(x,y,Pre^i(\mathcal{O})))$
    $\varphi_i \leftarrow \varphi_i + (v_1 - v_2)$
end for
$\varphi_i \leftarrow \frac{\varphi_i}{m}$
```

{v_1 and v_2 are the classifier's predictions for two instances, which are constructed by taking instance y and then changing the value of each feature which appears before the i-th feature in order $\mathcal{O}$ (for v_1 this includes the $i-$th feature) to that feature's value in x. Therefore, these two instances only differ in the value of the $i-$th feature.}

3.2.1 Error/speed tradeoff.

We have established an unbiased estimator of the contribution. Now we investigate the relationship between the number of samples we draw and the approximation error. For each φ_i, the number of samples we need to draw to achieve the desired error, depends only on the population variance σ_i^2. In practice, σ_i^2 is usually unknown, but has an upper bound, which is reached if the population is uniformly distributed among its two extreme values. According to Eq. (1), the maximum and minimum value of a single sample are 1 and -1, respectively, so $\sigma_i^2 \leq 1$. Let the tuple $\langle 1-\alpha, e\rangle$ be a description of the desired error restriction and $P(|\varphi_i - \hat{\varphi}_i| < e) = 1-\alpha$ the condition, which has to be fulfilled to satisfy the restriction. For any given $\langle 1-\alpha, e\rangle$, there is a constant number of samples we need to satisfy the restriction: $m_i(\langle 1-\alpha, e\rangle) = \frac{Z_{1-\alpha}^2 \cdot \sigma^2}{e^2}$, where $Z_{1-\alpha}^2$ is the Z-score, which corresponds to the $1-\alpha$ confidence interval. For example, we want 99% of the approximated contributions to be less than 0.01 away from their actual values and we assume worst-case variance $\sigma_i^2 = 1$, for each $i \in N$. Therefore, we have to draw approximately 65000 samples per feature, regardless of how large the feature space is. The variances are much lower in practice, as we show in the next section.

For each feature value, the number of samples $m_i(\langle 1-\alpha, e\rangle)$ is a linear function of the sample variance σ_i^2. The key to minimizing the number of samples is to estimate the sample variance σ_i^2 and draw the appropriate number of samples. This estimation can be done during the sampling process, by providing confidence intervals for the required number of samples, based on our estimation of variance on the samples we already took. While this will improve running times, it will not have any effect on the time complexity of the method, so we delegate this to further work. The optimal (minimal) number of samples we need for the entire explanation is: $m_{min}(\langle 1-\alpha, e\rangle) = n \cdot \frac{Z^2_{1-\alpha}\cdot\overline{\sigma^2}}{e^2}$, where $\overline{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n} \sigma_i^2$. Therefore, the number $n\cdot\overline{\sigma^2}$, where $\overline{\sigma^2}$ is estimated across several instances, gives a complete description of how complex a model's prediction is to explain (that is, proportional to how many samples we need).

A note on the method's time complexity. When explaining an instance, the sampling process has to be repeated for each of the n feature values. Therefore, for a given error and confidence level, the time complexity of the explanation is $O(n\cdot T(\mathcal{A}))$, where function $T(\mathcal{A})$ describes the instance classification time of the model on $\mathcal{A}$. For most machine learning models $T(\mathcal{A}) \leq n$.

4. Empirical Results

The evaluation of the approximation method is straightforward as we focus only on approximation errors and running times. We use a variety of different classifiers both to illustrate that it is indeed a general explanation method and to investigate how the method behaves with different types of classification models. The following models are used: a decision tree (DT), a Naive Bayes (NB), a SVM with polynomial kernel (SVM), a multi-layer perceptron artificial neural network (ANN), Breiman's random forests algorithm (RF), logistic regression (logREG), and ADABoost boosting with either Naive Bayes (bstNB) or a decision tree (bstDT) as the weak learner. All experiments were done on an off-the-shelf laptop computer (2GHz dual core CPU, 2GB RAM), the explanation method is a straightforward Java implementation of the equations presented in this paper, and the classifiers were imported from the Weka machine learning software (`http://www.cs.waikato.ac.nz/~ml/weka/index.html`).

The list of data sets used in our experiments can be found in Table 1. The first 8 data sets are synthetic data sets, designed specifically for testing explanation methods (see Robnik-Šikonja and Kononenko, 2008; Štrumbelj et al., 2009). The synthetic data sets contain the following concepts: conditionally independent features (CondInd), the xor problem (Xor, Cross, Chess), irrelevant features only (Random), disjunction (Disjunct, Sphere), and spatially clustered class values (Group). The Oncology data set is a real-world oncology data set provided by the Institute of Oncology, Ljubljana. To conserve space, we do not provide all the details about this data set, but we do use an instance from it as an illustrative example. Those interested in a more detailed description of this data set and how our previous explanation method is successfully applied in practice can refer to our previous work (Štrumbelj et al., 2009). The remaining 14 data sets are from the UCI machine learning repository (Asuncion and Newman, 2009).

Table 1: List of data sets used in our experiments. The variance of the most complex feature value and the variance of most complex model to explain are included.

model	# instances	# features (n)	max(σ_i^2)	max($\overline{\sigma^2}$)	$n\cdot$ max($\overline{\sigma^2}$)
CondInd	2000	8	0.25	0.06	0.48
Xor	2000	6	0.32	0.16	0.96
Group	2000	4	0.30	0.16	0.64
Cross	2000	4	0.43	0.14	0.92
Chess	2000	4	0.44	0.22	0.88
Sphere	2000	5	0.21	0.13	0.65
Disjunct	2000	5	0.10	0.06	0.30
Random	2000	4	0.19	0.12	0.48
Oncology	849	13	0.16	0.08	1.04
Annealing	798	38	0.08	0.02	0.76
Arrhythmia	452	279	0.03	10^{-3}	0.28
Breast cancer	286	9	0.22	0.10	0.90
Hepatitis	155	19	0.20	0.05	0.95
Ionosphere	351	34	0.20	0.04	1.36
Iris	150	4	0.23	0.10	0.40
Monks1	432	6	0.29	0.12	0.72
Monks2	432	6	0.31	0.27	1.62
Monks3	432	6	0.20	0.07	0.42
Mushroom	8124	22	0.24	0.05	1.10
Nursery	12960	8	0.22	0.03	0.24
Soybean	307	35	0.20	0.01	0.35
Thyroid	7200	21	0.18	0.02	0.42
Zoo	101	17	0.25	0.02	0.14

The goal of our first experiment is to illustrate how approximated contributions converge to the actual values. The fact that they do is already evident from the theoretical analysis. However, the reader might find useful this additional information about the behavior of the approximation error. The following procedure is used. For each data set we use half of the instances for training and half for testing the explanation method. For each dataset/classifier pair, we train the classifier on the training set and use both the explanation method and its approximation on each test instance. For the approximation method the latter part is repeated several times, each time with a different setting of how many samples are drawn per feature. Only synthetic data sets were used in this procedure, because the smaller number of features allows us to compute the actual contributions. Some of the results of this experiment are shown in Fig. 2. We can see that it only takes about a thousand of samples per feature to achieve a reasonably low approximation error. When over 10000 samples are drawn, all the contributions across all features and all test instances are very close to the actual contributions. From the discussion of the approximation error, we can see that the number of samples depends on the variance in the model's output, which in turn directly depends on how much the model has learned. Therefore, it takes only a few samples for a good approximation when explaining a model which has acquired little or no knowledge. This might be either due to the model not being able to learn the concepts behind the data set or because there are no concepts to learn. A few such examples are the Naive Bayes model on the Group data set (model's accuracy: 0.32, relative freq. of majority class: 0.33), the Decision Tree on Monks2 (acc. 0.65 , rel. freq. 0.67), and Logistic Regression on the Random data set (acc. 0.5 , rel. freq. 0.5). On the other

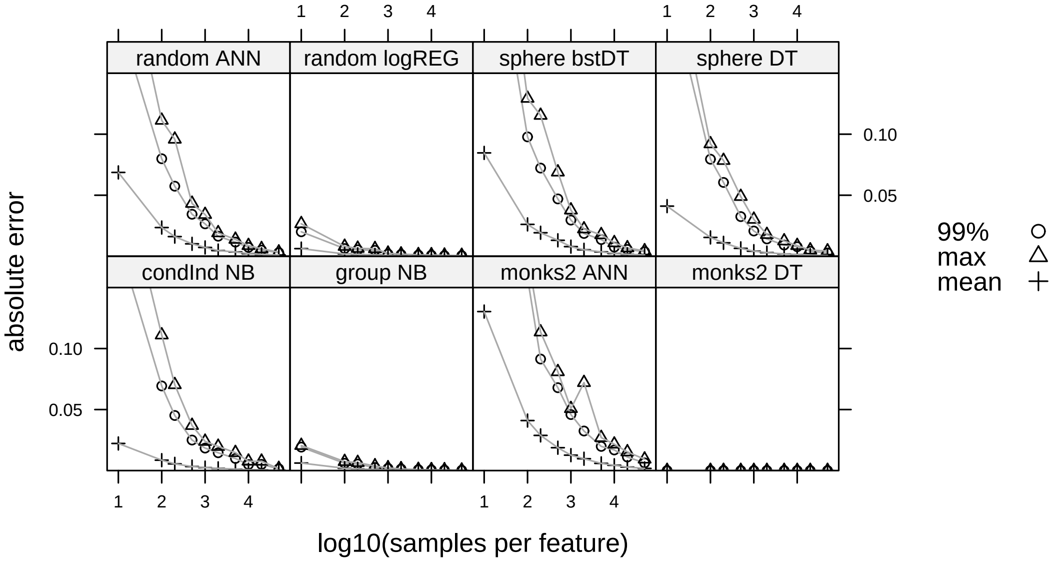


Figure 2: Mean, 99th-percentile, and maximum errors for several dataset/classifier pairs and across different settings of how many samples are drawn per feature. Note that the error is the absolute difference between the approximated and actual contribution of a feature's value. The maximum error is the largest such observed difference across all instances.

hand, if a model successfully learns from the data set, it requires more samples to explain. For example, Naive Bayes on CondInd (acc. 0.92 , rel. freq. 0.50) and a Decision Tree on Sphere (acc. 0.80 , rel. freq. 0.50). In some cases a model acquires incorrect knowledge or over-fits the data set. One such example is the ANN model, which was allowed to over-fit the Random data set (acc. 0.5 , rel. freq. 0.5). In this case the method explains what the model has learned, regardless of whether the knowledge is correct or not. And although the explanations would not tell us much about the concepts behind the data set (we conclude from the model's performance, that it's knowledge is useless), they would reveal what the model has learned, which is the purpose of an explanation method.

In our second experiment we measure sample variances and classification running times. These will provide insight into how much time is needed for a good approximation. We use the same procedure as before, on all data sets, using only the approximation method. We draw 50000 samples per feature. Therefore, the total number of samples for each dataset/classifier pair is: $50000n$ times the number of test instances, which is sufficient for a good estimate of classification times and variances. The maximum σ_i^2 in Table 1 reveal that the crisp features of synthetic data sets have higher variance and are more difficult to explain than features from real-world data sets. Explaining the prediction of the ANN model for an instance Monks2 is the most complex explanation (that is, requires the most samples - see Fig. 2), which is understandable given the complexity of the Monks2 concept

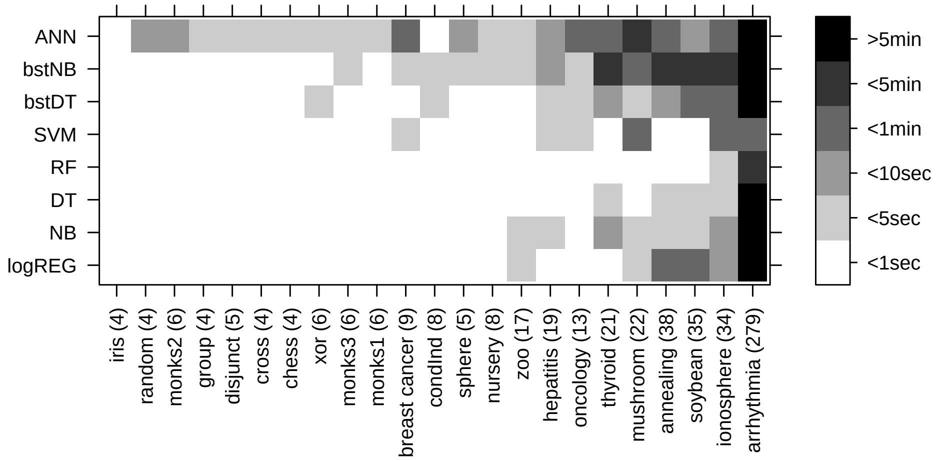


Figure 3: Visualization of explanation running times across all dataset/classifier pairs.

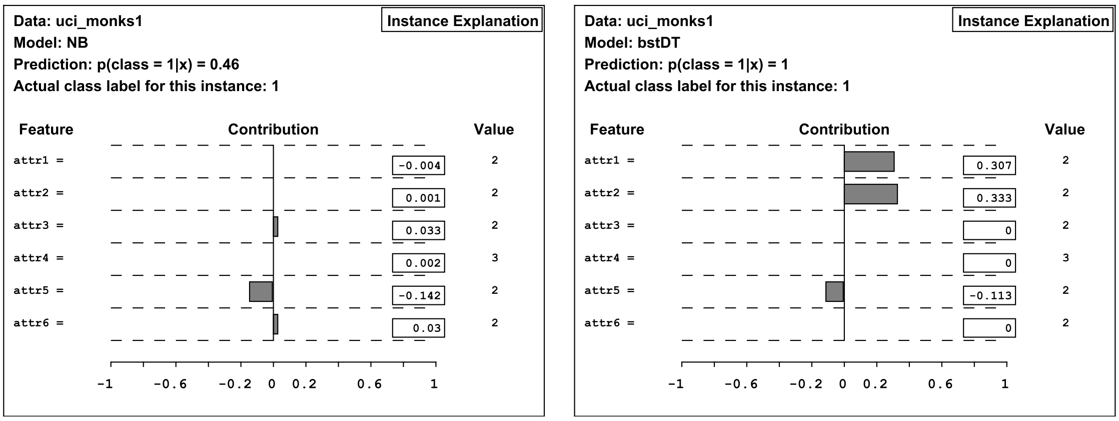
(class label = 1 iff exactly two feature values are 1). Note that most maximum values are achieved when explaining ANN.

From the data gathered in our second experiment, we generate Fig. 3, which shows the time needed to provide an explanation with the desired error $\langle 99\%, 0.01\rangle$. For smaller data sets (smaller in the number of features) the explanation is generated almost instantly. For larger data sets, generating an explanation takes less than a minute, with the exception of bstNB on a few data sets and the ANN model on the Mushroom data set. These two models require more time for a single classification.

The Arrhythmia data set, with its 279 features, is an example of a data set, where the explanation can not be generated in some sensible time. For example, it takes more than an hour to generate an explanation for a prediction of the bstNB model. The explanation method is therefore less appropriate for explaining models which are built on several hundred features or more. Arguably, providing a comprehensible explanation involving a hundred or more features is a problem in its own right and even inherently transparend models become less comprehensible with such a large number of features. However, the focus of this paper is on providing an effective and general explanation method, which is computationally feasible on the majority of data sets we encounter. Also note that large data sets are often reduced to a smaller number of features in the preprocessing step of data acquisition before a learning algorithm is applied. Therefore, when considering the number of features the explanation method can still handle, we need not count irrelevant features, which are not included in the final model.

4.1 Example Explanation

Unlike running times and approximation errors, the usefulness and intuitiveness of the generated explanations is a more subjective matter. In this section we try to illustrate the usefulness of the method's explanations with several examples. When interpreting the explanations, we take into account both the magnitude and the sign of the contribution.



(a) *NB model*

(b) *bstDT model*

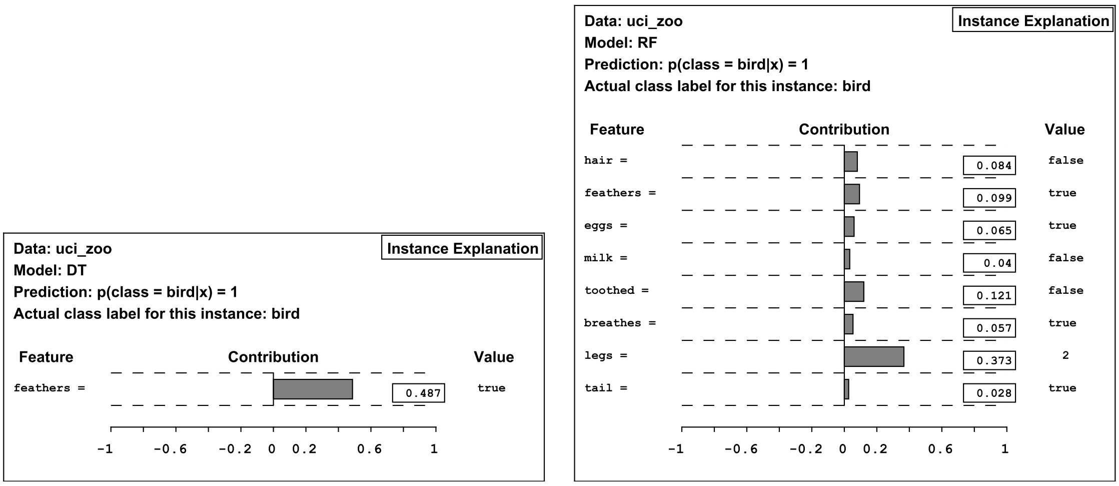
Figure 4: The boosting model correctly learns the concepts of the Monks1 data set, while Naive Bayes does not and misclassifies this instance.

If a feature-value has a larger contribution than another it has a larger influence on the model's prediction. If a feature-value's contribution has a positive sign, it contributes towards increasing the model's output (probability, score, rank, ...). A negative sign, on the other hand, means that the feature-value contributes towards decreasing the model's output. An additional advantage of the generated contributions is that they sum up to the difference between the model's output prediction and the model's expected output, given no information about the values of the features. Therefore, we can discern how much the model's output moves when given the feature values for the instance, which features are responsible for this change, and the magnitude of an individual feature-value's influence. These examples show how the explanations can be interpreted. They were generated for various classification models and data sets, to show the advantage of having a general explanation method.

The first pair of examples (see Fig. 4) are explanations for an instance from the first of the well-known Monks data sets. For this data set the class label is 1 iff attr1 and attr2 are equal or attr5 equals 1. The other 3 features are irrelevant. The NB model, due to its assumption of conditional independence, does not learn the importance of equality between the first two features and misclassifies the instance. However, both NB and bstDT learn the importance of the fifth feature and explanations reveal that value 2 for the fifth feature speaks against class 1.

The second pair of examples (see Fig. 5) is from the Zoo data set. Both models predict that the instance represents a bird. Why? The explanations reveal that DT predicts this animal is a bird, because it has feathers. The more complex RF model predicts its a bird, because it has two legs, but also because the animal is toothless, with feathers, without hair, etc... These first two pairs of examples illustrate how the explanations reflect what the model has learnt and how we can compare explanations from different classifiers.

In our experiments we are not interested in the prediction quality of the classifiers and do not put much effort into optimizing their performance. Some examples of underfitting and



(a) *DT model*

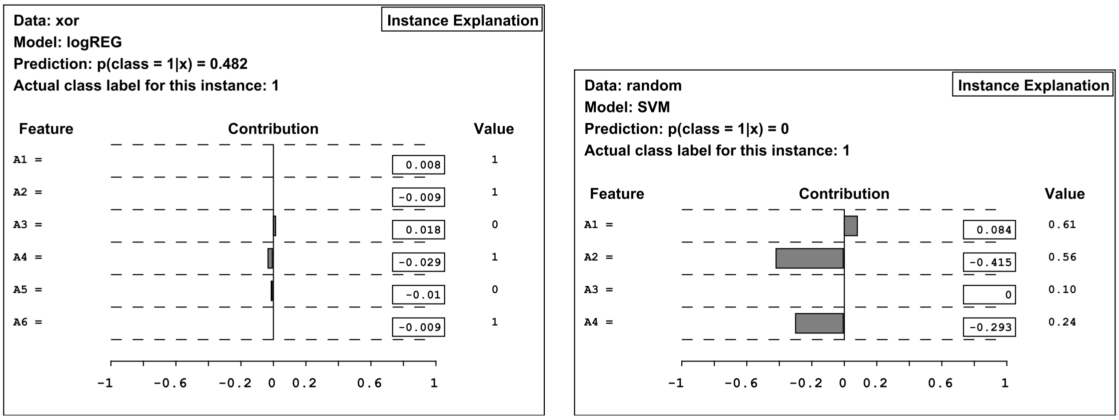
(b) *RF model*

Figure 5: Explanations for an instance from the Zoo data set. The DT model uses a single feature, while several feature values influence the RT model. Feature values with contributions ≤ 0.01 have been removed for clarity.

overfitting are actually desirable as they allow us to inspect if the explanation method reveals what the classifier has (or has not) learned. For example, Fig. 6(a) shows the explanation of the logREG model's prediction for an instance from the Xor data set. Logistic regression is unable to learn the exclusive-or concept of this data set (for this data set the class label is the odd parity bit for the first three feature values) and the explanation is appropriate. On the other hand, SVM manages to overfit the Random data set and finds concepts where there are none (see Fig. 6(b)).

Fig. 7(a) is an explanation for ANN's prediction for the introductory instance from the Titanic data set (see Fig. 1). Our explanation for the NB model's prediction (Fig. 7(b)) is very similar to the inherent explanation (taking into account that a logarithm is applied in the inherent explanation). The ANN model, on the other hand, predicts a lower chance of survival, because being a passenger from the 3rd class has a much higher negative contribution for ANN.

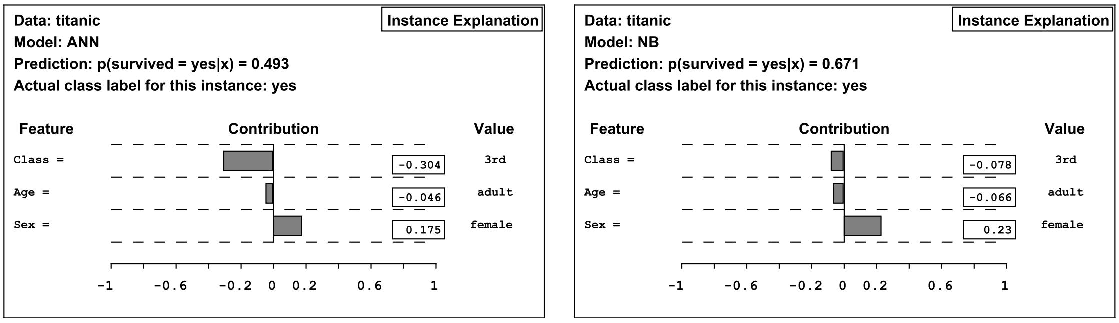
Our final example illustrates how the method can be used in real-world situations. Fig. 8 is an explanation for RT's prediction regarding whether breast cancer will (class = 1) or will not (class = 2) recur for this patient. According to RF it is more likely that cancer will not recur and the explanation indicates that this is mostly due to a low number of positive lymph nodes (nLymph). The lack of lymphovascular invasion (LVI) or tissue invasion (invasive) also contributes positively. A high ratio of removed lymph nodes was positive (posRatio) has the only significant negative contribution. Oncologists found this type of explanation very useful.



(a) *logREG model*

(b) *SVM model*

Figure 6: The left hand side explanation indicates that the feature values have no significant influence on the logREG model on the Xor data set. The right hand side explanation shows how SVM overfits the Random data set.



(a) *ANN model*

(b) *NB model*

Figure 7: Two explanations for the Titanic instance from the introduction. The left hand side explanation is for the ANN model. The right hand side explanation is for the NB model.

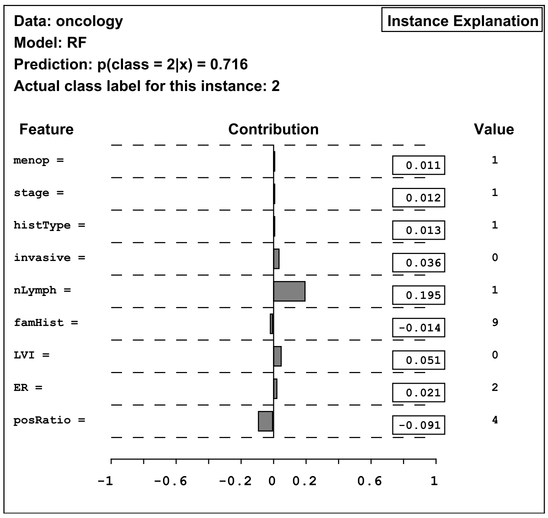


Figure 8: An explanation or the RF model's prediction for a patient from the Oncology data set.

5. Conclusion

In the introductive section, we asked if an efficient and effective general explanation method for classifiers' predictions can be made. In conclusion, we can answer yes. Using only the input and output of a classifier we decompose the changes in its prediction into contributions of individual feature values. These contributions correspond to known concepts from coalitional game theory. Unlike with existing methods, the resulting theoretical properties of the proposed method guarantee that no matter which concepts the classifier learns, the generated contributions will reveal the influence of feature values. Therefore, the method can effectively be used on any classifier. As we show on several examples, the method can be used to visually inspect models' predictions and compare the predictions of different models.

The proposed approximation method successfully deals with the initial exponential time complexity, makes the method efficient, and feasible for practical use. As part of further work we intend to research whether we can efficiently not only compute the contributions, which already reflect the interactions, but also highlight (at least) the most important individual interactions as well. A minor issue left to further work is extending the approximation with an algorithm for optimizing the number of samples we take. It would also be interesting to explore the possibility of applying the same principles to the explanation of regression models.

References

R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8:373–389, 1995.

Artur Asuncion and David J. Newman. UCI Machine Learning Repository, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 2009.

Barry Becker, Ron Kohavi, and Dan Sommerfield. Visualizing the simple bayesian classier. In *KDD Workshop on Issues in the Integration of Data Mining and Data Visualization*, 1997.

Leo Breiman. Random forests. *Machine Learning Journal*, 45:5–32, 2001.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers and Operations Research*, 2008. (in print, doi: 10.1016/j.cor.2008.04.004).

Shay Cohen, Gideon Dror, and Eytan Ruppin. Feature selection via coalitional game theory. *Neural Computation*, 19(7):1939–1961, 2007.

Lutz Hamel. Visualization of support vector machines with unsupervised learning. In *Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8. IEEE, 2006.

Aleks Jakulin, Martin Možina, Janez Demšar, Ivan Bratko, and Blaž Zupan. Nomograms for visualizing support vector machines. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 108–117, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X.

Alon Keinan, Ben Sandbank, Claus C. Hilgetag, Isaac Meilijson, and Eytan Ruppin. Fair attribution of functional contribution in artificial and biological networks. *Neural Computation*, 16(9):1887–1915, 2004.

Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence journal*, 97(1–2):273–324, 1997.

Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23:89–109, 2001.

Igor Kononenko and Matjaz Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms.* Horwood publ., 2007.

Vincent Lemaire, Raphaël Fraud, and Nicolas Voisine. Contact personalization using a score understanding method. In *International Joint Conference on Neural Networks (IJCNN)*, 2008.

David Martens, Bart Baesens, Tony Van Gestel, and Jan Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476, 2007.

Stefano Moretti and Fioravante Patrone. Transversality of the shapley value. *TOP*, 16(1):1–41, 2008.

Martin Možina, Janez Demšar, Michael Kattan, and Blaž Zupan. Nomograms for visualization of naive bayesian classifier. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 337–348, New York, NY, USA, 2004. Springer-Verlag New York, Inc. ISBN 3-540-23108-0.

Richi Nayak. Generating rules with predicates, terms and variables from the pruned neural networks. *Neural Networks*, 22(4):405–414, 2009.

Francois Poulet. Svm and graphical algorithms: A cooperative approach. In *Proceedings of Fourth IEEE International Conference on Data Mining*, pages 499–502, 2004.

Marko Robnik-Šikonja and Igor Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20:589–600, 2008.

Lloyd S. Shapley. *A Value for n-person Games*, volume II of *Contributions to the Theory of Games*. Princeton University Press, 1953.

Duane Szafron, Brett Poulin, Roman Eisner, Paul Lu, Russ Greiner, David Wishart, Alona Fyshe, Brandon Pearcy, Cam Macdonell, and John Anvik. Visual explanation of evidence in additive classifiers. In *Proceedings of Innovative Applications of Artificial Intelligence*, 2006.

Geoffrey Towell and Jude W. Shavlik. Extracting refined rules from knowledge-based neural networks, machine learning. *Machine Learning*, 13:71–101, 1993.

Erik Štrumbelj, Igor Kononenko, and Marko Robnik Šikonja. Explaining instance classifications with interactions of subsets of feature values. *Data & Knowledge Engineering*, 68 (10):886–904, 2009.