Arguments in Interactive Machine Learning

Martin Možina
Faculty of Computer and Information Science, University of Ljubljana, Slovenia
E-mail: martin.mozina@fri.uni-lj.si

Keywords: argumentation, interactive machine learning, argument-based machine learning

Received: November 7, 2017

In most applications of machine learning, domain experts provide domain specific knowledge. From previous experience it is known that domain experts are unable to provide all relevant knowledge in advance, but need to see some results of machine learning first. Interactive machine learning, where experts and machine learning algorithm improve the model in turns, seems to solve this problem. In this position paper, we propose to use arguments in interaction between machine learning and experts. Since using and understanding arguments is a practical skill that humans learn in everyday life, we believe that arguments will help experts to better understand the models, facilitate easier elicitation of new knowledge from experts, and can be intuitively integrated in machine learning. We describe an argument-based dialogue, which is based on a series of steps such as questions and arguments, that can help obtain from a domain expert exactly that knowledge which is missing in the current model.

Povzetek: V strojnem učenju je pridobivanje domenskega znanja pogosto prvi korak, ključen za definicijo učnih primerov, njihovih opisov in cilja učenja. Težava je, da eksperti večinoma niso sposobni dobro izraziti svojega znanja. Lažje je, če jim najprej pokažemo preliminarne, čeprav napačne rezultate strojnega učenja, saj eksperti tako lažje uvidijo, katero domensko znanje strojno učenje potrebuje. Postopek, kjer strojno učenje in ekspert izmenjaje izboljšujeta naučeni model, se imenuje interaktivno strojno učenje. V tem članku predlagamo uporabo argumentov v komunikaciji med računalnikom in ekspertom. Ljudje se argumentiranja naučimo zgodaj in ga veliko uporabljamo. Če bi računalniki znali svoje znanje predstaviti s pomočjo argumentov ter znali upoštevati človeške argumente pri svojem učenju, bi to vodilo do lažje komunikacije in posledično do bolj točnih in bolj razumljivih računalniških modelov. V članku pokažemo, kako vključiti argumentacijo v strojno učenje in opišemo ključna vprašanja ter odgovore v dialogu med strojnim učenjem in ekspertom, ki vodijo do tistega domenskega znanja, ki naučenemu modelu manjka.

1 Introduction

Domain experts are often involved in the development of a machine learning application. They help define the machine learning problem, provide learning examples, labels, and attributes of these examples. In some cases, they are even able to provide prior knowledge that is then incorporated into machine learning algorithms, which often results in more accurate and comprehensible models.

Acquiring domain knowledge is therefore one of the key tasks in machine learning, unfortunately a very difficult one, known also as the Feigenbaum knowledge acquisition bottleneck [4]. Domingos [2] identified several reasons why combining machine learning and expert knowledge often fails and how it should be approached. One of the reasons is that the results of machine learning are rarely optimal on the first attempt. An iterative improvement, where experts and computer improve the model in turns is needed. Furthermore, some knowledge is hard to make explicit. It turns out that humans are much better at explaining particular cases than eliciting general knowledge.

There are more and more machine learning studies using iterative improvements. Fails et al. [3] used the term *interactive machine learning* to describe an iterative system for correcting errors of an image segmentation system. Since then, researchers have presented many advantages of systems that allow users to interact with machine learning. Beside having better final performance, such as accuracy score, these works report that users also gain trust and understanding of their systems. A particularly interesting one was introduced by Stumpf [18], where a user can comment on automatically generated explanations provided by a learned model. These comments are then used as constraints in the system when relearning the model. Kulesza [9] called such an interaction *explanatory debugging*, because users identify "bugs" in a system by inspecting explanations and then explain necessary corrections back to the system.

We propose a similar approach that targets domain experts instead of end users. Explanatory debugging aims at building flexible applications, which can easily conform to the preferences of a user. In a spam filtering application, for example, an explanation might include the words that contributed to the prediction. When a user disagrees with the prediction, she can select some of these words and mark them as not being indicative of spam. The system must then

reduce the influence of these words in the future.

In our case, we focus on enabling the domain experts to elicit their knowledge in the development of a machine learning application. Our approach is less constrained, because experts can use general arguments to explain and to provide feedback back to machine learning. It seems that argument is the right tool for this problem, as humans have a lot of experience with arguments. We are using them every day to convince, negotiate, express and explain our opinions.

An argument in its simplest form is expressed as a set of premises that support a conclusion. In most cases, the link between premises and conclusion is not deductive, but presumptive. Consider, for example, the following argument:

Premise 1: Raising taxes increases government revenues.

Premise 2: Government needs money.

Conclusion: Government should raise taxes.

This argument is plausible, because raising taxes can increase revenues. It is also possible that it does not, if the taxes are already too high. However, when such an argument is put forward, involved parties understand that the conclusion might not be correct. If domain experts used arguments to express their knowledge, it would not be absolutely correct, however they would be able to express their domain knowledge more easily and in a natural way.

In this position paper we do not present any machine learning algorithms, experiments or results. Instead, we motivate the use of arguments in interactions between domain experts and machine learning. The motivation is based on the following two reasons: a) humans are already well practiced in argumentation and b) with some changes machine learning algorithms can communicate using arguments. In this paper, we focus on the second reason, since human argumentation is already well covered in the literature [19]. We identify what modifications of machine learning algorithms are needed to enable the use of arguments and which questions should we ask the domain experts to receive the most relevant information.

The main contribution of this work are instructions how to enable a general machine learning algorithm to use arguments. This includes presenting explanations in terms of arguments and the definition of the constraint that arguments impose on learning. In the previous work [14] we presented an actual implementation of learning rules from arguments. In this paper, this idea is generalized. Another contribution is a description of the refinement loop (a list of steps) for obtaining the most relevant knowledge from the domain expert. We have already presented several versions of this loop in our previous publications (see [14, 8]). Here, we unify these versions and provide more detailed explanations of the steps with practical examples. Finally, this paper motivates the use of arguments in interactive machine learning and supports this motivation with arguments.

2 Explaining classifications and arguments

Explaining decision or actions of intelligent systems to end users has many benefits [11]. It can positively affect the systems use, enable better understanding of the system and result in making people trust it.

Some machine learning models have the inherent capability of generating explanations, such as decision trees or classification rules [5]. Similarly, additive models, such as naïve Bayes or logistic regression, can use weights given to features to provide explanations of their decisions [15].

However, most of the contemporary machine learning research focuses on optimizing some abstract evaluation measure, such as classification accuracy or root mean squared error, and does not consider explanations at all. There have been some attempts to explain the decisions of of such methods. For example, Štrumbelj and Kononenko [17] suggested an algorithm for generating explanations of incomprehensible methods. They evaluate prediction importance of each feature by computing the difference between the classifier's prediction of an example and the prediction of the same example when this feature is omitted. This difference is then used in the explanation of the classifier's decision for this particular example.

We shall now define the relation between explanations in machine learning and arguments. We mentioned in the introduction that an argument contains a set of premises to support a conclusion. Since classification is the conclusion of the machine learning system, and the explanation contains the main reasons for this conclusion, it seems that an explained classification is already an argument.

Explanations of classifications rarely contain one argument only. Usually, an explanation provides reasons for and against the predicted class. For example, in the case of classification rules, we can present all rules covering the classifying example. Similarly, in nomograms [15] or in the general feature-based explanation framework [17], influences of features can be either positive or negative. Showing reasons for and against predicted class is beneficial to a domain expert, since it shows all relevant information that the underlying system used to infer a decision, which increases expert's understanding of the system [9].

An explanation thus contains arguments for and arguments against the predicted class value, without explaining the actual details of the algorithm for inferring the final decision. Knowing positive and negative factors is often sufficient for human understanding, as it is similar to how arguments are used in a human conversation. In a dialogue between two persons, arguments supporting one side are often challenged with the opposing arguments. It is not rare that the same set of arguments will lead to different conclusions of the participants in the dialogue, because they have different viewpoints and employ different internal reasoning mechanisms. Yet, knowing the opposing arguments is still beneficial, because they increase our understanding of the opposing viewpoints and therefore deepen our under-

standing of the issue. By analogy, it is more important for experts to understand which factors, both positive and negative, influenced the machine learning decision, and less how it was derived.

3 Argument-based machine learning

Argument-based machine learning (ABML) is a special case of learning from data and prior knowledge, where prior knowledge is represented with arguments [14]. A specific property of arguments is that they relate to a single example only and are not general as prior knowledge usually is. Several reviews and comparisons of different applications of prior knowledge are available, see for example [6, 12, 20].

The problem with domain knowledge occurs when experts are asked to provide general knowledge. Consider, for example, asking a physician to write down general rules for diagnosing pneumonia. A very difficult task. On the other hand, this physician can easily diagnose a certain patient and explain why he has pneumonia. For this reason we suggest using arguments to elicit and represent background knowledge. While asking experts to provide general background knowledge can be a difficult task, asking them to articulate their knowledge through arguments has proved to be much more efficient [2, 8].

In ABML, arguments are used to enhance learning examples. Each argument is attached to a single learning example, while one example can have several arguments. There are two types of arguments: positive arguments are used to explain (or argue) why a certain learning example is in the class as given, and negative arguments are used to explain why it should not be in the class as given. Examples with attached arguments are called *argumented examples*.

An ABML method needs to induce a model that will explain the classification of an example using the arguments provided by the expert. An ABML method, therefore, needs to be able to explain its decisions with arguments for and against, as we described in the previous section. Moreover, it needs to be able to accept input arguments and use these arguments in explanation, which allows the expert to immediately see the impact. The reasons from the positive arguments should become a part of the arguments for the class value in the explanation, and the reasons from the negative arguments should be mentioned within the against arguments in the explanation. Such explanation is therefore more comprehensible from the expert's perspective, since it uses the same terms as the expert [8].

For example, a diagnostic machine learning system might argue that a patient probably has pneumonia, because he is a male and he is coughing. A medical expert could then counter argue that this person has pneumonia, because he has high temperature. Then, ABML should induce a new model for automatic diagnosis, which would state high temperature (among others) as the reason for this particular patient with pneumonia.

Such instance-based constraints are different from how constraints are usually implemented in machine learning, because they relate to one example only. The system does not need to mention temperature in explanations of other examples, in fact, it could even mention low temperature in explanations of other patients with pneumonia, and that would still not violate the constraint.

Arguments are presumptive by nature and that is the main reason why arguments can not be applied generally, but to specific examples only. When a medical doctor explains a diagnosis of a patient, his or her argument contains many unstated premises that seemed unimportant at a time or were simply forgotten. Maybe fever is typical only for a certain type of pneumonia or only for a certain part of the population.

To implement an argument-based variant of a machine learning algorithm, one needs to take care that the arguments from experts are mentioned in explanations. This is easier achieved with models that are a composition of several parts. For example, an argument-based random forest could simply select only trees that are consistent with arguments. In our research group we implemented the ABCN2[1] algorithm [14], an extension of the CN2 algortihm [1], which learns classification rules from argumented examples. The main difference between the original CN2 and ABCN2 algorithms is in the definition of the covering relation. In the standard definition, a rule covers an example if the condition part is true for this example. In ABCN2, a rule covers an argumented example if the condition part is true and rule is consistent with positive arguments and not consistent with negative arguments.

4 A dialogue between a domain expert and a knowledge engineer

Although interactive machine learning assumes that end users (domain experts) directly interact with machine learning algorithms, we shall assume that a knowledge engineer acts as an intermediate between a domain expert and the algorithm, since some of the suggested steps in this section would be difficult to implement automatically.

Having a machine learning algorithm that can generate arguments and can accept expert's arguments, we will now define how a domain expert and a knowledge engineer can interact with arguments. We propose a series of moves that defines a dialogue between a domain expert and a knowledge engineer. A dialogue is a goal-directed conversation between two parties, in which parties are taking turns. In each turn, a participant makes a move that responds to the previous move. In this information-seeking dialogue, a knowledge engineer is trying to elicit relevant information from a domain expert by selecting relevant examples, using explanations of these examples, and asking the right questions. In our previous previous applications of ABML, we

[1]The latest version of ABCN2 can be found at `https://github.com/martinmozina/orange3-abml`.

called this the ABML refinement loop [8]. In this paper, we unify several versions of this loop and present it in the context of the ideas from the previous two sections. Furthermore, the descriptions of the steps cover many different situations where the dialogue might lead us to.

Given that arguments always relate to a single example, an engineer and an expert talk about one example at a time. As it is unlikely that experts will have time to discuss all learning examples, selecting relevant examples is important. We call these examples *critical examples*. A discussion about a single critical example has the following seven steps.

Step 1: Selecting a critical example

Critical examples are those learning examples that would have a considerable positive influence on the quality of the model if some arguments were provided. Initially, we took misclassified examples with the highest predictive error as critical examples [8]. However, in our recent experiments, we discovered that it is better to select prototypical misclassified examples, as examples with the highest error are more likely to be outliers and are therefore hard to explain. There are various algorithms available for obtaining prototypical examples, one option is to use clustering and take centers of these clusters [16].

It should be noted that this procedure misses a whole group of potentially critical examples: examples that are correctly classified, however the model produces incorrect or unacceptable explanations. Until now we have not yet found a good criteria for selecting such examples.

Step 2: Presenting the critical example to the expert

In this step, a critical example with explanation from the machine learning model is presented to the domain expert. As critical examples are misclassified by the current model, the current explanation is likely wrong. Then, the domain expert is asked the following question: "Why is this example in the class as given?" The answer to this question should not contain the reasons that are mentioned in the current machine's exlanation.

Example. In one of the first applications of ABML, the goal was to distinguish between a good and a bad bishop in a chess position [13]. The learning data contained chess positions with one bishop only. Instances were described with attributes that are typically used in chess evaluation functions and each instance was classified as *bad bishop* or *good bishop*. One of the descriptive attributes was *mobility*, which counted the number of possible moves for the bishop. The algorithm initially learned that good bishops have high mobility. The first critical example was a position with a good bishop, which was blocked by a knight and was therefore not able to move (had low mobility). The expert was thus asked: "Why is the black bishop in this position good if it has low mobility?"

Step 3: The expert provides arguments for the critical example

The domain expert needs to provide at least one argument (a set of reasons) why the example's class value is as given. The argument must contain at least one reason, which was not in the original explanation provided by the machine learning method, otherwise this argument will not influence learning. If the expert can not give such an argument, we have to return to step 1 and provide another critical example.

In our previous experiments, a domain expert was unable to provide an argument due to the following two reasons. In the first case, an expert might find the example an outlier, because he or she cannot explain why the example is in this class. We can then remove the example from the data set, or, if not, prevent this example to become a critical example again. In the other case, which is also quite common, the expert discovers an error in the data. For example, it might turn out that the label of the example is wrong or that there is an error in the value of one of the descriptive attributes. Then we have to correct the error and start with another critical example.

Example. We used ABML to learn a diagnostic model for distinguishing between different types of tremor in patients with a neurological disease [7]. The patients were classified as *essential tremor* or *parkinsonian tremor* or *mixed tremor* (having both). In most cases, the expert (a physician) was able to explain critical examples. In one of the critical cases, however, the expert realized that some strong symptoms were overlooked at the time of diagnosis and, after a careful deliberation, decided to change the class value. In another critical case, the expert could not provide an argument, because the value of the attribute containing qualitative assessment of a physician had a incorrect value. After the value was corrected, the example was not critical anymore.

Step 4: Adding arguments to the critical example

A domain expert usually expresses arguments in natural language without considering domain description of the learning data set. The knowledge engineer then needs to rephrase provided arguments using domain description language (attributes).

However, expert's reasons in arguments are sometimes not covered with the current set of attributes. A domain expert often implicitly refers to an attribute missing from the current set of attributes. A knowledge engineer then needs to implement the new attribute, or change the definition of an old one. When the expert refers to unavailable attributes, which can not be added into the domain, we need to continue with another critical example.

Explaining examples with arguments has shown to be an effective tool for suggesting new attributes, since domain experts do not need to explicitly propose a set of relevant attributes, but can implicitly suggest new attributes in explanations.

Example. In the case with the bishop, the expert responded that the bishop's mobility is not limited, because the blocking knight can easily move to another square. We therefore had to redesign the mobility attribute by considering only pawns as blocking pieces. In the tremor application, the expert also suggested several new attributes. When a patient with essential tremor was selected as critical, the expert mentioned the presence of harmonics (a certain pattern in drawings of patients) as a clear signal of essential tremor. There was no attribute in the domain that would explicitly define the presence of harmonics. However we could compute a new boolean attribute (from four existing attributes) representing whether the harmonics were present or not.

Step 5: Discovering counter examples

After arguments are added to the critical example, ABML relearns the model. Arguments often apply to many other examples, and not just to the critical example, therefore these arguments will be mentioned in explanations of other examples. When these examples come from the same class as the critical example, such behavior is not problematic, it is in fact favorable, since more examples are now explained using the expert terms.

On the other hand, if the model uses positive arguments of the critical example to explain examples from the opposite class, we should check the validity of these explanation with the expert. A *counter example* is an example from the opposite class that is consistent with the positive argument provided by the expert, the induced model mentions this positive argument in the explanation of this counter example, and the inclusion of this argument in the data resulted in a higher prediction error for this example (e.g. the example is now misclassified or has a higher probabilistic error).

Example. After attaching the above argument to the bishop critical example, it turned out that high mobility is not enough, as a position with a bad and highly mobile bishop turned out as the counter example. The provided argument was consistent with the counter example (the mobility of the bishop was high), however it was from the opposite class (the bishop was bad).

Step 6: Refining arguments using counter examples

When a counter example was found, the expert needs to revise the initial arguments with respect to the counter example. The expert is now asked "Why is critical example in one class and why counter example in the other?" The expert may now revise the original argument and explain the difference between these two examples. The procedure then returns to the previous step and seeks for more counter examples.

Example. Comparing critical and counter positions in the chess example, the expert decided that the counter example had a noticeably worse pawn structure. This reason was added to the original argument of the critical example. Therefore, the initial argument (high mobility) was extended with an argument specifying good pawn structure. Afterwards, no counter examples were found.

Step 7: Pruning arguments with similar examples

In argumentation, to make an argument stronger and less susceptible to counter-arguments, humans often provide more reasons that are actually needed. In ABML, however, too many reasons will result in poor generalization.

As the last step in the discussion of the particular critical example, we should evaluate reasons in the provided argument whether they are necessary. A reason is unnecessary when its removal a) does not negatively affect the prediction accuracy of the critical example, b) does not introduce new counter-examples, and c) generalizes the argument to *similar examples*. Given a reason and an argument, a critical example is similar to another example, when they are from the same class and the argument would also apply to the similar example if it was removed. A single similar example is then shown to the expert, who needs to decide whether the same argument could be used for both examples.

Example. Although we encountered too specific arguments in almost every application of ABML, we have not yet used pruning. For example, when we tried to classify student Prolog programs as correct or incorrect [10] and the expert was asked to provide arguments for a correct critical program, he would often mention many syntactical patterns that are indicative of a correct program. After evaluating the rules that were learned from these arguments, we found out that many of the mentioned reasons were redundant. Therefore, in that application pruning would lead to a simpler and less fragmented model.

The above seven steps are repeated until the system can not find any new critical examples or some goal (such as accuracy or comprehensibility of the model) is achieved.

5 Conclusion

When a knowledge engineer is faced with a machine learning problem, she usually needs to first sit down with a domain expert and try to define the problem. As mentioned in the paper, this process is not trivial, since experts usually can not give us all the answers in advance, but an interactive process is preferred. Even with an interactive process, the communication can still be difficult, when domain experts do not understand machine learning, and knowledge engineers do not understand the domain.

In this paper, we proposed to use arguments as a communication method for bridging the gap between domain experts and machine learning. Argumentation is a skill used in everyday communication that everyone learns to a certain extent. Therefore, if machine learning results and domain experts' knowledge were represented as arguments, it would facilitate smoother communication.

We first showed how machine learning can interact with domain experts by explaining its decisions using arguments

for and against. Such explanations resemble argumentative reasoning and should thus be good enough for experts. Afterwards, we demonstrated how experts can express their knowledge by explaining particular learning examples with positive and negative arguments. The learning algorithm then uses these arguments to guide learning towards a model that is consistent with data and provided arguments. Finally, to close the loop, we described a dialogue between a domain expert and a knowledge engineer designed to drive the expert to provide useful knowledge.

When we first presented the ABML idea [14], the arguments were only used to explain learning examples. In one of the following experiments [13], we defined the ABML refinement loop, where arguments turned out to be an effective tool for elicitation of new attributes. This refinement loop was then further revised through many applications [8]. In this paper, we presented an extended version of the ABML refinement loop, where communication between a domain expert and a domain engineer comprises of several questions and arguments. This involves machine generated arguments, asking expert to give counter-arguments to machine learning arguments, and refining arguments given counter examples and similar examples.

Acknowledgement

This work was partly supported by the Slovene Agency for Research and Development (ARRS). We would also like to thank the two anonymous reviewers for valuable suggestions on this paper and colleagues from the Artificial Intelligence Laboratory, who contributed a lot in the past in the development of the ABML idea.

References

[1] Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *Machine Learning - Proceeding of the Fifth Europen Conference (EWSL-91)*, pages 151–163, Berlin, 1991.

[2] Pedro Domingos. Toward knowledge-rich data mining. *Journal of Data Mining and Knowledge Discovery*, 15:21–28, 2007.

[3] Jerry Alan Fails and Dan R. Olsen, Jr. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 39–45, 2003.

[4] Edward A. Feigenbaum. Knowledge engineering: the applied side of artificial intelligence. In *Proc. of a symposium on Computer culture: the scientific, intellectual, and social impact of the computer*, pages 91–107, New York, NY, USA, 1984. New York Academy of Sciences.

[5] Alex A. Freitas. Comprehensible classification models - a position paper. *SIGKDD Explorations Newsletter*, 15(1):1–10, 2014.

[6] Valerio Grossi, Andrea Romei, and Franco Turini. Survey on using constraints in data mining. *Data Mining and Knowledge Discovery*, 31(2):424–464, 2017.

[7] Vida Groznik, Matej Guid, Aleksander Sadikov, Martin Možina, Dejan Georgiev, Veronika Kragelj, Samo Ribari, Zvezdan Pirtoek, and Ivan Bratko. Elicitation of neurological knowledge with argument-based machine learning. *Artificial intelligence in medicine*, 57(2):133–144, 2013.

[8] Matej Guid, Martin Možina, Vida Groznik, Aleksander Sadikov, Dejan Georgijev, Zvezdan Pirtoek, and Ivan Bratko. Abml knowledge refinement loop: A case study. In *Proceedings of the 2012 IEEE 20th International Symposium (ISMIS 2012)*, pages 41–50, 2012.

[9] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 126–137, 2015.

[10] Timotej Lazar, Martin Možina, and Ivan Bratko. Automatic extraction of ast patterns for debugging student programs. In *International Conference on Artificial Intelligence in Education*, pages 162–174. Springer, 2017.

[11] Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2119–2128, 2009.

[12] Violeta Mirchevska, Mitja Lustrek, and Matjaz Gams. Combining domain knowledge and machine learning for robust fall detection. *Expert Systems*, 31:163–175, 2014.

[13] Martin Možina, Matej Guid, Jana Krivec, Aleksander Sadikov, and Ivan Bratko. Fighting knowledge acquisition bottleneck with argument based machine learning. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 234–238, 2008.

[14] Martin Možina, Jure Žabkar, and Ivan Bratko. Argument-based machine learning. *Artificial Intelligence*, 171(10/15):922–937, 2007.

[15] Martin Možina, Janez Demšar, Michael Kattan, and Blaž Zupan. Nomograms for visualization of naive bayesian classifier. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Knowledge Discovery in Databases: PKDD*

2004: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, September 20-24, 2004. Proceedings, pages 337–348, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[16] J. Arturo Olvera-López, J. Ariel Carrasco-Ochoa, J. Francisco Martínez-Trinidad, and Josef Kittler. A review of instance selection methods. *Artif. Intell. Rev.*, 34(2):133–143, 2010.

[17] Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18, March 2010.

[18] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. Interacting meaningfully with machine learning systems: Three experiments. *Int. J. Hum.-Comput. Stud.*, 67(8):639–662, 2009.

[19] Douglas Walton. *Foundamentals of Critical Argumentation; 1st edition*. Cambridge University Press, 2005.

[20] Ting Yu. *Incorporating prior domain knowledge into inductive machine learning: its implementation in contemporary capital markets.* PhD thesis, University of Technology, Sydney. Faculty of Information Technology., 2007.