






# Classification Rules Explain Machine Learning

Matteo Cristani<sup>1</sup><sup>a</sup>, Francesco Olivieri<sup>2</sup><sup>b</sup>, Tewabe Chekole Workneh<sup>1</sup><sup>c</sup>, Luca Pasetto<sup>1</sup><sup>d</sup>  
and Claudio Tomazzoli<sup>3</sup><sup>e</sup>

<sup>1</sup>Department of Computer Science, University of Verona, Italy

<sup>2</sup>School of Computer Science, Griffith University, Brisbane, Australia

<sup>3</sup>CITERA, University of Rome, Italy

**Keywords:** Machine Learning, eXplainable AI, Approximation, Anytime Methods.

**Abstract:** We introduce a general model for explainable Artificial Intelligence that identifies an explanation of a Machine Learning method by *classification rules*. We define a notion of *distance* between two Machine Learning methods, and provide a method that computes a set of classification rules that, in turn, approximates another black box method to a given extent. We further build upon this method an anytime algorithm that returns the best approximation it can compute within a given interval of time. This anytime method returns the minimum and maximum difference in terms of approximation provided by the algorithm and uses it to determine whether the obtained approximation is acceptable. We then illustrate the results of a few experiments on three different datasets that show certain properties of the approximations that should be considered while modelling such systems. On top of this, we design a methodology for constructing approximations for ML, that we compare to the no-methods approach typically used in current studies on the explainable artificial intelligence topic.

## 1 INTRODUCTION


The reproduction of results obtained by up-to-date methods of Machine Learning (ML) and Deep Learning (DL) approaches, in a fashion that could be understood by humans is one of the emerging topics in recent Artificial Intelligence studies. The overall idea of explainable Artificial Intelligence lies on the possibility of doing what is devised above.


If we look at the concepts underlying this very notion, we observe a relevant distance between how a ML, and in specific case, a DL system executes the classification and the natural way in which human beings identify *rules* that execute similar analyses on the same data. This discrepancy has been remarked in numerous cases, and is one of the reasons why explainable artificial intelligence has emerged.


When looking at a set of data, we can regard them as *expression* of implicit regularity, the discover of


which is indeed an intelligent process, and even a *logical base* for an induction process to learn those regularities. Given that we expect data to grow over time, this process can be performed in two models:

- The data of a classification system have at least a partition of human - made labels. Therefore there is the so called *ground truth*: a training set and a test /validation set can be determined, while we always have an execution dataset. After a training and validation phase over the training and test sets, an algorithm can try and predict the values against the execution dataset by looking at two or more observation instants. This method is based on the existence of a ground truth, so we name this *grounded model*.
- The data of a classification system do not have any human - made labels. We need another classification algorithm (maybe a previously evaluated one) so that a performance comparison can be made over the execution dataset by looking at two or more observation instants. This method does not provide ground to the conclusions it takes, it only validates a model against another one. Therefore we name this model *cross-validation model*.

<sup>a</sup> <https://orcid.org/0000-0001-5680-0080>

<sup>b</sup> <https://orcid.org/0000-0003-0838-9850>

<sup>c</sup> <https://orcid.org/0000-0001-5756-2651>

<sup>d</sup> <https://orcid.org/0000-0003-1036-1718>

<sup>e</sup> <https://orcid.org/0000-0003-2744-013X>

The basic idea of this paper is that a classification system  $C$  explains another classification system  $C'$  when two conditions hold:

- $C$  is an approximation of  $C'$ , i.e., the way in which  $C$  forecasts the results is close to the way in which  $C'$  does so, and
- $C$  is a readable rule-based systems. We will provide a non-formal definition of this concept of readability in the following paragraphs.

To better identify the underlying concept, we provide here a toy example about the basic notion of explainability.

**Example 1.** Consider a binary classification system that we can only observe as an input-output black box technology, as illustrated in Figure 1. Assume that

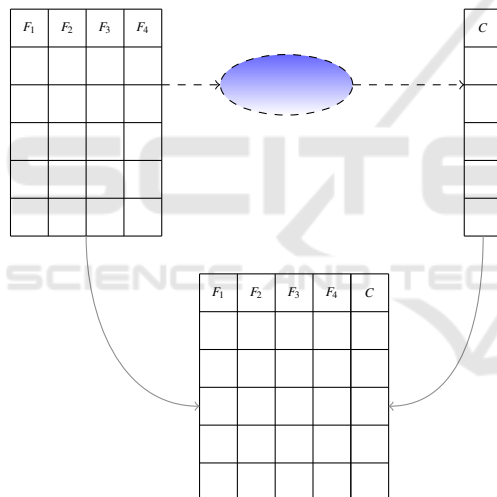


Figure 1: A black box input-output data configuration.

the rows in the table under analysis of the observed classifier respond to one association rule  $R1$  and to an association rule  $R2$ , each (for the sake of simplicity) with a positive error of 3% and a negative error of 2%. The two rules establish

- **$R1$**  when the values in column  $F1$  are greater than a threshold value limit  $t_{F1}$  and the values in column  $F2$  are less than a threshold value  $t_{F2}$  then the classification value on column  $C$  is true;
- **$R2$**  when the values in column  $F3$  are greater than a threshold value  $t_{F3}$  and the values in column  $F4$  are less than a threshold value  $t_{F4}$  then the classification value on column  $C$  is true.

Now, the rule system  $\{R1, R2\}$  is an approximation of the black box that has a positive error of 3% and a negative error of 2%.

Based on the idea illustrated above, we build a model of explanation that can be summarised as follows: explaining means approximating with the constraint that an approximation is better when it is simpler. Therefore, to devise a good technology for explaining a black-box system, we need to identify a system that is readable and a hierarchy among readable systems that are as simpler and more accurate (precise or recalling) as possible.

The rest of this paper is organised as follows. Section 2 introduces some basic notions used in the rest of the paper, and Section 3 discusses an algorithmic approach based on the introduced notions. Section 5 reviews relevant literature, and Section 6 takes conclusions and sketches further work.

## 2 BASIC DEFINITIONS

We introduce, from scratch, some basic notions that we are about to use in order to introduce a notion of explainability that is to be formalised in the rest of this paper. There is a large part of readers who might consider this section very introductory. However, it is important to get to the ground notions to highlight how a notion of actual explainability should be introduced.

**Definition 1.** Assume a collection of three datasets  $\langle Train, Test, Exec \rangle$  where: (i) the training set  $Train$  and the test set  $Test$  are formed by the same  $n + 1$  columns, and (ii) the execution set  $Exec$  is formed by the first  $n$  columns of training and test sets. We name these  $n$  columns the classification features and the  $(n + 1) - th$  one the class.

The classification problem consists in defining a method that learns the class classification to the classification features from the training set, verifies it on the test set to provide its measures of accuracy, precision and recall and finally applies the classification to the execution test, where we expect the behaviour of the classifier will respect the measures taken on the test set.

A system like the one just devised above is said to be a classifier. Every classifier is intrinsically described by its confusion matrix that is a  $k$ -entry matrix establishing the relative account of elements (frequency) in each class obtained for each value of the classification by the classifier performed on the test set. When the classifier is binary (i.e., the classes are only two), the matrix cells are named true positive ( $TP$ ), false positive ( $FP$ ), true negative ( $TN$ ),

and *false negative (FN)*, indicating the elements: (i) *correctly classified* (true positive and true negative), and (ii) *incorrectly classified* (false positive and false negative). In general, when looking at the confusion matrix we also use the acronyms  $P = TP + FN$  for positive elements,  $N = TN + FP$  for negative elements,  $T = TP + FP$  for elements classified correctly,  $F = FP + FN$  for elements classified incorrectly and finally  $A = TP + FP + TN + FN$  for the size of test set (namely *all* elements. We henceforth limit ourselves in studying binary classifiers.

Accuracy is the frequency of a classifier correct answers. It is defined as

$$a(C) = \frac{TP + TN}{A} = \frac{T}{A} = 1 - \frac{F}{A}$$

Precision measures the probability of the classifier's positive answers to be correct. In other terms, the more a classifier is precise, the fewer are the positive incorrect answers it gives. It is defined by

$$p(C) = \frac{TP}{TP + FP} = \frac{TP}{T}$$

Recall measures the probability of the classifier's negative answers to be false. In other terms, the higher is a classifier's recall, the more are the positive correct answers it gives. It is defined by

$$r(C) = \frac{TP}{TP + FN} = \frac{TP}{P}$$

Once we have trained a classifier on the training set, we can measure its behaviours in terms of quality of the classification on the test set.

We can now introduce a measure that has been used in various applications of classification rules, and in general applies to rule-based systems including decision trees. To do so, we start by formalising the notion of classification rules.

A *simple* value constraint may be either an equality constraint, where a feature is set to be valued in a given way, an inequality constraint, that is the opposite of the above, and a minority/majority constraint when it imposes the value of the feature to span below/over a given value on the range of admissible values for that feature. A minority/majority constraint can be either weak (while including the extreme value) or strong (while excluding the extreme value). When the values of a feature are not ordered, then the equality/inequality constraints are the only admissible ones. Boolean expressions on simple value constraints are hence *value constraints*.

**Definition 2.** *An classification rule is a classifier that consists of a test of value constraints on the features for a classification problem that forecasts the class of that problem.*

It is not difficult to show that every value constraint on an ordered domain can be represented by a finite collection of intervals and semi-intervals, along with a finite set of inequalities (Stergiou and Koubarakis, 2000). Conversely, when a discrete unordered and finite domain is involved, the largest number of elements involved in a value constraint is exactly the size of the domain minus 1. For instance, if a finite domain is formed by three elements, then admissible constraints are 7, corresponding to the request to range over a subset of the 3 elements.

R1 of Example 1 is a classification rule. An *classification rule* system is a set of rules, that are to be applied *disjunctively*, and assumed to validate one single class. The set  $\{R1, R2\}$  of Example 1 is a classification rule system. An element of the test or execution set is classified by means of a rule system when it is valued by at least one of the rule system in that class.

At a very simple level, we can introduce a notion of complexity of a rule. A rule *length* is the sum of intervals or the sum of disjoint values for unordered and finite domains, for each feature.

**Example 2.** *Consider a feature F1 ranging on natural numbers, and a feature F2 ranging on real numbers. A rule consists in the expression:*

$$((1 \leq F1 \leq 12) \vee (14 \leq F1 \leq 33)) \wedge ((F2 \neq 3.14) \wedge (F2 \geq 2.72)) \rightarrow True$$

The length of the above rule is 4, because the constraints appearing in it are formed in turn by two pairs of constraints on two different features, both of length two. Clearly, when a rule  $R$  has length  $l_1$  and a rule  $R'$  has length  $l_2$ , then  $l_1 \leq l_2$  is to be read  $R$  is shorter than  $R'$ .

Two rules are in the containment order  $\subseteq$  ( $R1 \subseteq R2$ ) *iff* when  $R2$  is satisfied then  $R1$  is satisfied. For instance, consider a rule  $R'$ :

$$((1 \leq F1 \leq 12) \vee (14 \leq F1 \leq 33)) \rightarrow True$$

and a rule  $R''$ :

$$((1 \leq F1 \leq 12)) \rightarrow True$$

$R'$  and  $R''$  in the relation  $R' \subseteq R''$ . In fact, when  $R''$  is satisfied,  $R'$  is satisfied as well. A rule  $R$  is said to be *simpler* than a rule  $R'$  when  $R$  is shorter than  $R'$  and  $R' \subseteq R$ . Extensively, a rule set  $S_1$  is simpler than a rule set  $S_2$  when each rule of  $S_1$  is simpler than at least one rule in  $S_2$ .

We can provide an absolute measure of complexity of a classification rule system by computing its *volume* a measure inspired by classic *software* metrics: the product of maximum length of a rule in the system and number of rules. This notion of volumes

presents two primary advantages that we need to consider in this phase.

When two systems have the same volume they can be rather different but may be reasonably similar in terms of the *effort* we need to devise them. The computation of rules is expensive both in terms of required design effort, and in terms of the computational effort to build the rules. Both effort of design and computational cost depend on the length of the rules and on the number of rules. We could have considered *length* of the rule system as the *sum of lengths of the rules*. However, these would have made equivalent two systems in a way that is in fact independent of the number of rules. This is actually different from what we expect to be a *good* explanation. In fact, an explanation should be understandable, and therefore we expect it to be made of *simple rules* and that these rules are not too many. Obviously we should get a tradeoff, for the ideal explanation would result, from the above premises, the empty rule system. Therefore, volume is a better measure of length, for it could accommodate, in the same volume, or roughly the same, a number of possible different explanations that are similar in terms of design effort, and let us able to choose the *best* ones, by considering the *most accurate* or precise or recalling.

To summarise, the idea is that when two approximations are comparable in terms of *simplicity* we should choose the most accurate (precise, recalling), and on the opposite, when two approximations are comparable in terms of *accuracy* (respectively precision, recall) we should choose the simpler one.

We can shift our attention to the details of the approach, by devising a method to compute approximations for a black box classifier. Consider a classifier  $C$  that we cannot look more in detail than by its behaviour, namely by looking at the results of the classification on an execution test, without knowing how the results are obtained. The idea of the method is to perform the classification on a subset of the execution set, that we can control. On the input/output behaviour of the classifier we can build a training set, that is used to extract a classification rule set. We are now able to build a confusion matrix and use it to devise the correct understanding of the behaviour of the classifier.

Specifically, when such a method is computed, we can state that a binary classifier  $C$  has been approximated by a classification rule system  $S$  with accuracy  $\alpha$ , precision  $\pi$  and recall  $\rho$  (by using the measures obtained by the confusion matrix) with volume  $V$ . Ideally, we should also derive the idea that a rule set is a *better* approximation of a given classifier when it has the same accuracy (respectively precision, re-

call) but a smaller volume, or the same volume but a better accuracy (respectively precision, recall). Obviously, the idea that accuracy, precision and recall remain the same (either alone or in triple) is unrealistic, for these values are intrinsically unstable on classification rule setup. We can establish a range interval on which accuracy, precision and recall are to be considered equivalent, that is the *confidence* on that operation. The confidence is thus a value  $\chi$  such that when accuracy (respectively, precision, recall) of a set  $S_1$  is to be considered *roughly the same* of accuracy (respectively, precision, recall) of a set  $S_2$  is that because  $|\alpha(S_1) - \alpha(S_2)| \leq \chi$  (and correspondingly for  $\pi$  and  $\rho$ ).

Therefore, we can state that a rule set  $S_1$  is a better approximation of a rule set  $S_2$  because  $S_1$  has less volume than  $S_2$  and they are ordered by better accuracy (respectively, precision, recall) or at most roughly the same.

### 3 ALGORITHMS FOR EXPLAINING THE BLACK BOX

Given the analysis discussed above, we can devise a method to compute *optimal rule systems* that approximates a classifier, or, on more practical design of the method, a rule system that approximates a classifier with an acceptable accuracy (respectively precision, recall). Consider an approximation  $S_1$ . We can make two kinds of operations. The former consists in looking for admissible simplifications that preserve accuracy (respectively precision, recall) and reduce volume. The latter, on the opposite, looks for improvement on the accuracy (respectively precision, recall) but preserve volume.

At this point of this discussion, we need to specify an important aspect of the concept of rule system we have devised so far. The idea of disjunctive systems is quite simple, and it is so for we can provide room for constraints on the range by means of a method that results polynomial on deterministic machines. The majority of methods like Apriori as well as methods for similar approaches to other rule system, as in, for instance, decision trees, provide incremental computation of the rule system. Thus, a rule system is more complex and potentially more accurate, precise and recalling, then the rule system computed on the previous step by those methods. To obtain good explanations we therefore should look at these methods with a critical eye: the purpose of the computation is to choose among the *partial* solutions obtained by these methods that we consider *good* explanations, more than *good* classifiers. In fact, we do not really have



a comparison with the ground truth, for the only object we look at is a classifier that acts in the realm of the execution set. We should manage a choice of the quality of the explanation by *looking* at the relative accuracy (respectively precision, recall) but we cannot ensure that the approximation obtained *only introduces* the errors we measure on the relative comparison, for new errors could have been introduced, as well as errors made by the explained classifier are incorporated as well (and obviously we can also have the opposite, inverted effect, where errors of the explained classifier are *corrected* by the explanation).

The above reasoning can be described as the core of the methodology we propose here. Let us summarise the concept we are looking at:

1. We are given a black box classifier  $C$ , for which we aim at computing an acceptable explanation  $E$ ;
2. We assume a *partition* of the domain classified by  $C$  in two subsets *Train* and *Test*, where the classes (positive and negative) are assigned;
3. We use the subset of the execution set *Train* to devise a collection of *rule classification systems*;
4. We choose among the systems computed in the above Step 3 an explanation that has both a *good* volume, and a *good* accuracy (respectively precision, recall).

Naturally, computing *every possible* system is unaffordable, for even if the number might be finite, is likely to be unacceptably large. A possible way to overwhelm the mentioned drawback is to limit to a method that works on incremental algorithm

At first, as an initial perspective, we evaluate the opportunity of applying the devised approach. Consider a feature  $F1$  ranging over an infinite range of values. Given a set of records on which the training set sets the value of the row to true, we can collect the values in total order by ordering algorithm in  $O(n \cdot \log(n))$ . Once we have devised the correct ordering, the line of values can be marked by the class elements true, and therefore partitioned into open intervals.

For instance, assume that we have  $F1$  valued  $a_1, a_2, \dots, a_5$  on  $n$  rows, and that  $a_1$  and  $a_2$  are valued true, while  $a_3$  is valued false, and  $a_4$  and  $a_5$  are both valued true. The interval partition is  $(F1 \leq a_2) \vee (a_2 < F1 < a_4) \vee (F1 \geq a_4)$ . Each single rule generated in this way matches the training set with no errors, but other disjunctive rules can be generated with errors as well. For instance, in particular for discrete cases, we may have situations in which a value is attributed to a feature and that feature has

some true and some false with that value. The *majority* method chooses the most frequent among the two cases, the *unanimity* method chooses true only when all the cases unanimously give a true. Mix strategy lies on the notion that instead of basic majority we require qualified one to reinforce the method (that is equivalent to mitigating the unanimity by requiring a significant percentage instead of 100%). Clearly, generating rules on single feature does not improve the results significantly, for they could only be considered separately, and there is an extensive literature on how to combine them that has shown to work only in specific and limited conditions.

All the above notions are quite well-known from Decision Tree approaches as well as association rules. The novelty we introduce here is the idea that approximation is computed by using one of these approaches, and then evaluated by means of the notions of relative accuracy (respectively precision, recall) and total volume as a measure of the quality (simplicity) of the approximation. If we thus consider an approximation obtained by a rule set  $S$  we can state that  $S$  is a good approximation when it is the simplest, or when it has the best relative accuracy. Clearly the two quantities are in tradeoff. It is very likely that the simplest explanation would not result to be the most relatively accurate, and the way around. If we start with an Apriori method, we can simply generate the fit rules one at a time, and evaluate the resulting rule set in terms of volume. Classic implementations of Apriori allow us to do so, by invoking it on the basis of the step to be performed. Once we have invoked the algorithm, we can obtain the values of the resulting approximation. During this process we can keep information on the computed model by enumerating the results along with the accuracy (respectively precision, recall) and the volume. At the end of the Apriori execution we can generate the partially ordered set of these approximations and choose the best candidate, namely among the ones with optimal volume that with best accuracy (respectively precision, recall). The volume and accuracy (respectively precision, recall) could be specified with a parameter of rough equivalence,  $\chi_V, \chi_\alpha(\chi_\pi, \chi_\rho)$  that is passed to the algorithm itself.

## 4 EXPLANATION ALGORITHMS

The basic explanation method we introduce is exactly the Apriori algorithm, that is trapped in each step of the refinement. Apriori algorithm. This is introduced in Algorithm 1.

Algorithm 1: Apriori algorithm.

---

```

1:  $L_1 \leftarrow \text{Frequent1-itemset}$ 
2:  $k \leftarrow 2$ 
3: while  $L_{k-1} \neq \emptyset$  do
4:    $Temp \leftarrow \text{candItemSet}(L_{k-1})$ 
5:    $C_k \leftarrow \text{freqOfItemSet}(Temp)$ 
6:    $L_k \leftarrow \text{complItemSetWithMinSup}(C_k, \text{minsup})$ 
7:    $k \leftarrow k + 1$ 
8: end while
9: return L

```

---

With a little abuse of notation we assume here that the algorithm we implement invokes Apriori *until step*  $k$ , and left it suspensively waiting for another invocation (as in trap/interrupt parallelism methods). Therefore we shall name this invocation by *Apriori(k)*. Algorithm 2 considers the measure  $\lambda$ , that could be accuracy, precision or recall.

Substantially, Algorithm 1 is an *anytime* version of the classic mining method for association rules. We introduced this concept here, for three distinct reasons.

- Firstly, we need a method that could be devised to perform in practice. Every technique of association rule generation is computationally expensive, and since the computation of explanations is here *explorative*, for we need to understand, at least in principle, what explanation is the best, or, at least, whether an explanation is acceptable, there is a serious risk of not being able to conclude the analysis within an even not particularly restrictive practical time limit;
- Secondly, Apriori is a very basic approach, and allow us to provide the second step of the investigation, described in 6, where *human in the loop* analysis is to be performed in order to validate with a gold standard the introduced notion of *volume*;
- Thirdly, for we look at an explainability concept that can be used *even when we do not know* what features have been extracted (as in the deep learning approach) and aim at devising a very general methodology for rule-based systems.

One main drawback of the above devised method is the limited ability that it has to refine the results. In fact, by definition of the Apriori method, every step increases the coverage given by the rules, and on the same time it simplifies the rules, by generating a classifier that has a smaller or identical volume. Therefore, the probability that the computed solution is computed just at the end of the execution of the algorithm is very high being the stability of the method very unusual. The introduction of the  $\chi$  values. The

Algorithm 2: Computing volume and accuracy algorithm.

---

```

1:  $Stack \leftarrow \emptyset$ ;
2:  $k \leftarrow 1$ 
3: while Apriori is not terminated, Increment  $k$  do
4:   Invoke Apriori( $k$ );
5:   Compute Volume and  $\lambda$  of  $L_k$ ;
6:   push Index of  $L_k$ , volume,  $\lambda$ ;
7: end while
8: return  $Stack$  optimal by (Volume,  $\lambda$ ) extensively
   considering the volume under the constraint of
   range equivalence given by  $\chi_V$  and accuracy (re-
   spectively precision, recall) under the appropriate
   parameter  $\chi_\lambda$  (with  $\lambda = \alpha, \pi, \rho$ )

```

---

second, well known drawback, regards the types of the data, that need to be discrete and finite. In fact, with infinite value ranges the predictive capability of Apriori algorithm is very basic, while with Decision Trees we can have much better results, from this viewpoint. However, while Apriori is definitely interruptible, by means of the invocation parameter  $k$ , as shown above, it does not make any sense to interrupt a DT method, for the computation of the tree could just be stopped over in the wrong deepening. The only case in which this can be done is when the method used to generate the tree starts from the root, and goes, bread-width, downward. This is the case of ID3. Again we could imagine to substitute the invocation to Apriori( $k$ ) by the invocation to ID3( $k$ ) where  $k$  denotes the *level* of the Decision Tree generated so far, we obtain the very same method of Algorithm 2.

The usage of ID3 could, potentially, improve the performances, while letting the underlying process of explanation intact. In the future we shall explore much more deeply methods that are based upon logic programming, such as Inductive Logic Programming, and Answer Set Programming.

To summarise the approach we consider two explanations ordered when they are similar in terms of volume and one is better than the other one in terms of accuracy (respectively precision or recall), or on the opposite, when they are similar in terms of accuracy (respectively precision, recall) but one has a smaller volume. Measuring volume is very easy for association rules, and can be extended in a straightforward way to decision trees. The very same notion cannot be applied to black-box systems.

Let us consider a simple abstract example. Assume that we aim at approximating a black-box system  $C$ . We can describe by  $\langle Train, Test, Exec \rangle$  the split of the input to  $C$  observed in a period of observation (data are randomly chosen to belong to Train and Test). After this period, we can feed Algorithm 2 with Train set and compute alternative Apriori so-

lutions to approximate the black-box. At top of the stack at the end of the process we shall have an Apriori set of rules that exhibit a good volume, in the sense that this will be the best (minimum) volume obtained by the anytime performance of the Apriori method.

While comparing two elements in the stack, we may have either a better element that has less rules, or a better element that has short maximum rule. On the other side of the evaluation, when two Apriori solutions are comparable in terms of volume, we may choose the best one in terms of accuracy (precision, recall). Notice that even a limited (within a given threshold) loss in terms of accuracy (precision, recall) or in terms of volume could be accepted as fruitful. In fact, when the Apriori anytime method devised above acts, we do not really know what is computed in terms of rules, for we can drop rules based on the invocation by Algorithm 2. Essentially, as it is devised, the method could use volume as an *heuristic* for a search in the solution space in a potential implementation based on efficient search algorithms such as SMA\*.

Note that the interruptible method is based on the very useful property of Apriori to discover rules *in order of length*. Therefore, dropping a rule requires a specific method. We have experimented two different possible conditions:

- We may drop rules that have *limited coverage*. This is a good idea in general, for they contribute only a little to the usefulness of the approach. However, typically, rule coverage decreases with length, and therefore the dropping will not be particularly effective;
- We may drop rules based on their relationship with *shorter rules*. For instance we do not need to generate a rule that has a superset of antecedents with respect to another rule already selected;

## 5 RELATED WORK

The long path that is bringing us from black box technologies, as for instance deep learning ones, to explainable solutions, has been only recently carried out. Moreover, there is an open debate on the actual features that we expect from an explainable system. First of all, we have in mind two crucial points:

- Explainable systems should be readable by humans, but also should be usable in order to identify the meaning of the machine learning activity that is performed;
- Explainability is both an issue of the design of a system, and of the interpretation of them.

What are thus explainability it twofold: good explanations for existing systems, and methodologies to design new systems that behave in an explainable way. On the way of this path we should consider the explainability as a basic property. There has been a recent focus on this very basic issue: how can we *measure* explainability? Answers to this question are central in the process of making our effort.

In a high level perspective, the study of Pedrycz et al. (Pedrycz, 2021) has provided a specific viewpoint on a variety of possible approaches to be employed for explainability, while discussing, in particular, the basic notion of explanation as a means to guarantee coordination of different learning methods.

Very general viewpoint on explainability can be found in the survey and also comparative study provided by Molnar et al. (Molnar et al., 2020) who reviewed critically the approaches developed so far on the argument of approximation for explainability. On research specific issues, we refer to the path of research conducted by Mereani et al. since 2019's preliminary study (Mereani and Howe, 2019) and further deepening (Mereani and Howe, 2021). For what concerns deep learning explanations Soares et al. have provided a comprehensive approach that could be considered the counterpart of the neural network investigation of Mereani et al. (Soares et al., 2021).

Association rules have been specifically investigated as means for explaining black boxes by Moradi et al. (Moradi and Samwald, 2021).

## 6 CONCLUSIONS AND FURTHER WORK

The promising results of this investigation are mainly due to the intuition that a rule-based system can be measured to perform better in terms of *explainability* than a black-box one. In fact, deep learning methods exhibit high accuracy, precision and recall for many real-world cases in which *perception* is involved, but when *reasoning* is involved than many aspects that cannot be considered in deep learning take place.

First of all, as amplily argued in many studies concerning *hybrid reasoning* (see, for instance, (Cristani et al., 2018b; Cristani et al., 2018a)), the connection between reasoning and decision making is strict, and therefore data coming from observations, including those related to perception are only a part of the decision process, for the knowledge we do have in front of the perception influences our conclusions. Secondly, practical usages of methods derived from Logic can incorporate that knowledge in the decision process, whilst perception-driven methods cannot.

Further developments of this study will be focusing on three aspects: (1) the applicability of logic programming methods, that some of the authors have been dealing with before (Cristani et al., 2015; Cristani et al., 2014; Olivieri et al., 2015; Cristani et al., 2016b) when connected to machine learning problems, (2) the usage of Inductive Logic Programming approaches as in (Lisi, 2008; Lisi, 2010; Lisi and Straccia, 2013), and (3) the study of applications in specific mining domains, including in particular social network and the web, for process mining purposes (Cristani et al., 2016a; Cristani et al., 2016c).

One important step that we shall carry out rather immediately aims at deploying an experiment with human subjects to validate the notion of volume as a predictor, along with performance indices like accuracy, precision and recall, of the judgment on explanations for a black-box system.

## REFERENCES

- Cristani, M., Domenichini, F., Olivieri, F., Tomazzoli, C., and Zorzi, M. (2018a). It could rain: Weather forecasting as a reasoning process. volume 126, pages 850–859.
- Cristani, M., Fogoroasi, D., and Tomazzoli, C. (2016a). Measuring homophily. volume 1748.
- Cristani, M., Karafili, E., and Tomazzoli, C. (2014). Energy saving by ambient intelligence techniques. pages 157–164.
- Cristani, M., Karafili, E., and Tomazzoli, C. (2015). Improving energy saving techniques by ambient intelligence scheduling. volume 2015-April, pages 324–331.
- Cristani, M., Olivieri, F., and Tomazzoli, C. (2016b). Automatic synthesis of best practices for energy consumptions. pages 154–161.
- Cristani, M., Olivieri, F., Tomazzoli, C., and Zorzi, M. (2018b). Towards a logical framework for diagnostic reasoning. *Smart Innovation, Systems and Technologies*, 96:144–155.
- Cristani, M., Tomazzoli, C., and Olivieri, F. (2016c). Semantic social network analysis foresees message flows. volume 1, pages 296–303.
- Lisi, F. (2008). Building rules on top of ontologies for the semantic web with inductive logic programming. *Theory and Practice of Logic Programming*, 8(3):271–300.
- Lisi, F. (2010). Inductive logic programming in databases: From datalog to DL+log. *Theory and Practice of Logic Programming*, 10(3):331–359.
- Lisi, F. and Straccia, U. (2013). Dealing with incompleteness and vagueness in inductive logic programming. volume 1068, pages 179–193.
- Mereani, F. and Howe, J. (2019). Exact and approximate rule extraction from neural networks with boolean features. pages 424–433.
- Mereani, F. and Howe, J. (2021). Rule extraction from neural networks and other classifiers applied to xss detection. *Studies in Computational Intelligence*, 922:359–386.
- Molnar, C., Casalicchio, G., and Bischl, B. (2020). Interpretable machine learning – a brief history, state-of-the-art and challenges. *Communications in Computer and Information Science*, 1323:417–431.
- Moradi, M. and Samwald, M. (2021). Post-hoc explanation of black-box classifiers using confident itemsets. *Expert Systems with Applications*, 165.
- Olivieri, F., Cristani, M., and Governatori, G. (2015). Compliant business processes with exclusive choices from agent specification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9387:603–612.
- Pedrycz, W. (2021). Design, interpretability, and explainability of models in the framework of granular computing and federated learning.
- Soares, E., Angelov, P., Costa, B., Castro, M., Nagesh Rao, S., and Filev, D. (2021). Explaining deep learning models through rule-based approximation and visualization. *IEEE Transactions on Fuzzy Systems*, 29(8):2399–2407.
- Stergiou, K. and Koubarakis, M. (2000). Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117.