Applied Mathematics & Information Sciences
*An International Journal*

http://dx.doi.org/10.18576/amis/100442

# On Post-Processing the Outputs of Prediction Systems: Strategies, Empirical Evaluations and a Case Study in Computer Security

*Mouaad Kezih*[1,*], *Mahmoud Taibi*[1], *Salem Benferhat*[2] *and Karim Tabia*[2]

[1] LASA Laboratory, Electronics Department, Faculty of Sciences, Badji Mokhtar University, P.O. Box 12, 23000 Annaba, Algeria.
[2] CRIL - CNRS UMR 8188, Artois University, France.

Received: 10 Apr. 2014, Revised: 20 Mar. 2015, Accepted: 23 Mar. 2015
Published online: 1 Jul. 2016

**Abstract:** Supervised classification is a well-known task in data-mining and it is widely used in many real world domains. Classifiers are automatic prediction systems used to predict the class label of items described by a set of features. In many areas, it is important to take into account some extra knowledge and constraints in addition to the one learnt or encoded by the classifier. In this paper, we propose an approach allowing to exploit the available domain knowledge with the predictions of a classifier. More precisely, we propose to post-process the predictions of a classifier in order to take into account some domain knowledge. This approach can be applied with any classifier be it probabilistic or not. We propose post-processing criteria and methods to encode and exploit different kinds of domain knowledge. Finally, the paper provides extensive experimental studies on a representative set of benchmarks and classification problems including imbalanced datasets. We also provide a case study on two crucial problems in computer security which are intrusion detection and alert correlation. Interestingly enough, the results show that using only some available knowledge about the training datasets or the performances of the used classifiers can improve these classifiers' efficiency while fitting the available domain knowledge.

**Keywords:** Classifiers, Post-processing, Revision, Computer Security, Intrusion Detection System (IDS)

## 1 Introduction

In real world applications, many problems are dealt with as classification tasks or more generally as prediction problems. Classification (also called supervised classification) is a well-known task in data-mining and machine learning. It consists in predicting the class of an object given its features. Examples of well-known classifiers are decision trees [32], Bayesian networks [19], SVMs [15], *k*NN [1], etc. Classifiers are predictive models built either from expert knowledge or automatically learnt from empirical data. Most works in classification deal either with learning efficient classifiers from data or combining multiple classifiers [25]. Many related issues receive also much interest especially regarding learning classifiers from imbalanced datasets [12], classifier evaluation, reject and drift options [13], non-exclusive or multiple class classification problems, etc.

In this paper, with deal with a new and complementary issue aiming to exploit any extra domain knowledge by post-processing the classifier predictions. Indeed, in many applications it is important to take into account some extra knowledge, constraints or preferences of the users. In computer security for instance, an operator monitoring and checking the alerts raised by intrusion detection systems [3], may want to select only 10% of most reliable alerts. The problem dealt with in this paper is the one of revising the predictions of a classifier in order to fit the user requirements. These latter can be seen as constraints to satisfy and can refer to expert knowledge on the addressed problem, preferences, etc. We addressed this problem originally in [6] in a computer security context and we dealt only with probabilistic classifiers. Moreover, in that work we proposed only two basic criteria to revise the predictions of a classifier. In the following, we mention mostly classifiers but the proposed approach can apply as well on any prediction or detection system as illustrated in our

* Corresponding author e-mail: kezih.mouad@gmail.com

case study in computer security.

In this paper, we address the problem of post-processing the predictions of a classifier in order to exploit any available domain knowledge. The main contributions of this paper are:

1. We propose a formalization of the problem of post-processing the predictions of a classifier in order to fit some extra knowledge. This problem is new in the supervised classification community and there is no formal definition for it.
2. We propose new post-processing criteria. In particular, we propose a criterion allowing to relabel the items where the classifier's confidence is low measured in terms of entropy. Another criterion is tailored for cost-sensitive classification problems and allows to choose the items to relabel based on the classifier confidence and the miss-classification costs.
3. We generalize and extend the post-processing procedure to any classifier and any prediction system instead of only probabilistic-based ones.
4. We carried out an extensive experimental study covering most of the problems dealt with in classification tasks. In particular, we evaluated on many benchmarks with different characteristics in terms of features number, instances, number of classes (namely binary and non binary classification problems). We also provide experimental results on the class imbalance problem.
5. We provide a case study on two typical computer security problems where it really makes sense to revise the predictions of a prediction/detection system with the users' domain knowledge, constraints and preferences.

This paper is organized as follows: In Section 2, we present the motivations of this work and review the related works. Section 3 gives insights into classifiers' predictions while Section 4 presents the strategies to post-process the predictions in presence of domain knowledge. In Section 5 we present the post-processing criteria. In Section 6 we present our experimental studies. In Section 7 we provide a case study in intrusion detection and alert correlation areas. Finally, Section 8 concludes the paper.

## 2 Related works and motivations

In supervised classification, many issues are still hot research topics and represent an active research field. For instance, a lot of interest and effort is devoted for designing efficient classifiers and for combining classifiers to take advantage of their complementarities and strengths. Approaches trying to exploit some kinds of domain knowledge mainly do it in a pre-processing step (for instance by choosing high quality training datasets, selecting good priors, etc.). Indeed, most works aiming to exploit background and expert knowledge along with classifiers focus on improving model learning and model selection. For example, in [39] the authors combine background knowledge elicited from experts and empirical data to better learn the structure of Bayesian networks.

As for post-processing a probabilistic classifier predictions, one can list the combination techniques where several classifiers are combined to exploit their mutual complementarities [34]. Note that multiple classifier combination [24,34] is concerned with aggregating the predictions made by multiple classifiers but there is no domain knowledge that is used for the combination. Other works dealing with classifier predictions are those based on the reject option where a prediction is made only if the probability of making a good decision is higher than a user defined threshold [13]. Note that the reject option relies only on the confidence of the classifier when making predictions and the user defined thresholds. This corresponds to another type of expert knowledge (the required confidence level by the user). Note also that a lot of works dealt with calibrating the posterior probability estimates [20] but such works aim to provide more reliable posterior probability distributions for the items to classify without considering any expert knowledge. Calibration is important for ranking predictions, combining multiple classifiers or when using the reject option.

In many real world applications, typically a classifier or a prediction/detection model is used to classify items of interest. For example, spam filters, intrusion detection systems [3], object, action and activity recognition systems in video analysis [38] are well-known detection/prediction models and in such domains, the models are not necessarily learnt from training data. Then if a user wants that his model complies to some specific requirements (for instance constraints or preferences) then he cannot learn a new model or tune the existing one. Our approach can well fit such needs and it is appropriate for both machine learning-based classifiers and prediction/detection models.

### *2.1 Domain knowledge*

The goal of our revision-based post-processing is to exploit the available extra-knowledge in order to fit the user's knowledge, constraints and preferences. In the following, we provide some typical domain knowledge a user may want to exploit over a classifier predictions:

– **i) Domain knowledge about the items to classify:** Assume that we have $n$ objects to classify denoted $o_1$, $o_2$,.., $o_n$. Then one may have information (in general or within a specific situation) that the amount of items of a class $c_i$ is greater than $c_j$ (namely, the probability

$p(c_i)>p(c_j)$). For example, in anomaly detection problems [3] which can be viewed as a classification task, it is common to assume that the frequency of *normal* events is greater than *abnormal* ones. Then one may want to satisfy all the time or in particular cases a constraint of the form *p(normal)>p(abnormal)* where *p(normal)* (resp. *p(abnormal)*) denotes the frequency of items detected as *normal* (resp. *abnormal*).

–**ii) A user's requirements:** In many prediction/detection applications, a user may want to have a specific amount of instances in a given class. For example, in computer security [3] and video surveillance applications [38], human operators monitoring the detected events are overwhelmed with the huge numbers of anomalous events and they are incapable to analyze them all. What is generally done in practice is to limit the number of alerts. This objective can be achieved by selecting among all the predictions a user specified amount that they can analyze. Such requirements represent application constraints or simple user preferences.

In the following, we use the generic term *domain knowledge* to designate the available knowledge of the application domain under consideration as well as the specific constraints and preferences of the users. Typically, one can have three types of domain knowledge that can be exploited to post-process the predictions of a classifier:

–*Knowledge about a single class:* This knowledge can be in the form of an amount or a frequency. For instance, a user may want to select exactly 100 top instances of class $c_i$ or select 2% of the items that belong to a class $c_i$.

–*Knowledge about the ranking over the classes:* In this case, a user may just want to have more or less instances of class $c_i$ than class $c_j$. This knowledge can be expressed for example as $p(c_i)>p(c_j)$. One may also want to have a complete ranking over the classes $p(c_i)>p(c_j)>..>p(c_k)$.

–*Knowledge about the class distribution:* The third kind of knowledge can be a precise distribution for all the predictions. Namely, for $i$=1..$k$, we specify the frequency $p(c_i)$ of items that should be predicted in class $c_i$.

Knowledge about the class distribution is the most exhaustive and accurate domain knowledge. In the experimental studies, we provide experiments using these three kinds of domain knowledge.

## 3 Classification and classifiers

### *3.1 Classification*

Classification, also known as supervised learning, consists in predicting the right class of an item. For example, spam filtering can be seen as a classification problem since the problem consists in classifying any new mail in one of predefined classes (namely *spam* or *normal*). In computer security, intrusion detection can also be seen as a classification task consisting in labeling the analyzed activities as *authorized* or *attack*).
Formally, a classification problem is defined by:

–*A feature space:* A set of attributes $A_1$, $A_2$,.., $A_n$ where each variable $A_i$ is associated with a domain $D_i$ which can be discrete or continuous. The set of attributes $A_1$, $A_2$,.., $A_n$ are observable and describe the objects to classify.

–*A class space:* It consists of a discrete variable $C$ with a domain $D_C$=$\{c_1, c_2, .., c_k\}$. The values $c_i \in D_C$ are called class instances or class labels.

A classifier is a function that associates a class $c_i \in D_C$ with an objet $a_1a_2..a_n$. This latter is an instantiation of the attributes $A_1$, $A_2$,.., $A_n$. The objective is to minimize a loss (or a miss-classification) function. Namely, a classifier aims to minimize the classification error rate. In cost-sensitive classification problems, the aim is to minimize the overall miss-classification cost.

### *3.2 Classifier outputs*

Classifiers are predictive models that can be grouped according to the nature of their outputs mainly into three categories:

–**Single class output:** Such classifiers only output the predicted class. An example of such classifiers is standard decision trees [32]. Some prediction and deception systems such as intrusion detection systems are of this type.

–**Ranking-based output:** This kind of classifiers output a ranking of the different class instances for the item to classify then one can select the first or the $n$ best candidate classes.

–**Score-based output:** It is the most informative output a classifier can provide allowing to predict and assess the classifier confidence regarding its predictions. Examples of probabilistic classifiers are Bayesian network classifiers.

## 4 Post-processing a classifier's predictions to fit domain knowledge

### *4.1 Post-processing strategies*

As illustrated on Figure 1, the objective is to design a post-processor to revise the predictions made by a classifier to fit the set of requirements of the user.

Assume that we have a set of items to classify denoted $\mathcal{O}$=$\{(a_1a_2..a_n)_1, (a_1a_2..a_n)_2 .. (a_1a_2..a_n)_m\}$ where

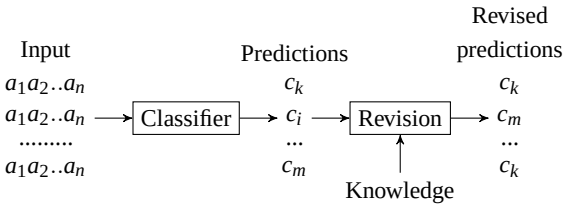


**Fig. 1:** Post-processing a classifier's predictions

$a_1a_2..a_n$ is an instantiation of the attributes $A_1A_2..A_n$. The classifier $f$ will associate with each instance $(a_1a_2..a_n)_i$ a class instance $c_k \in D_C$, denoted $c_k=f((a_1a_2..a_n)_i)$. Without loss of generality, let us assume that the classifier $f$ outputs a vector of scores $v_i=(s_1,s_2,..,s_k)_i$ for each instance to classify $(a_1a_2..a_n)_i$ (here, $k$ denotes the number of class instances, namely $k=|D_C|$). The score vector $(s_1,s_2,..,s_k)_i$ is :

i) *A posterior probability distribution* in case of using a probabilistic classifier. For instance, Bayesian network classifiers [19] associate with any object to classify a posterior probability distribution $v=(p(c_1|a_1..a_n),p(c_2|a_1..a_n),..,p(c_k|a_1..a_n))$. In the $k$-NN classifier, the scores $s_i$ could be the proportion of training items labeled $c_i$ among the $k$ selected items while classifying the item in hand. Generally, the score $s_i$ can be interpreted as the confidence, uncertainty or membership degree of the classifier that the right class is $c_i$.

ii) *A vector of zeros and ones* in case of classifiers outputting only class labels as predictions. For example, a classifier predicting $c_1$ will output the vector $(1,0,..,0)$ where the value 1 denotes the predicted classes while the remaining zeros exclude the corresponding classes. Well-known example of classifiers outputting only class labels is standard decision trees [32].

iii) *A probability distribution* to encode the ranking such that if $c_i$ is ranked before $c_j$ then $p(c_i)>p(c_j)$. It is easy to build a probability distribution $p$ over the class variable domain $D_C$ inducing the desired class ranking [20].

Note that there are calibration techniques [20] that can be used to scale and normalize any classifier outputs into a probability distribution. Using normalized probability distributions offers many advantages [20] for post-processing tasks such as prediction combination in multiple classifier systems, cost-sensitive classification, classification with reject option, etc.

In this paper, we deal with post-processing the predictions of a classifier where a prediction $c^*$ for an item $a_1..a_n$ is generally obtained according to the following rule:

$$c^* = argmax_{i=1..k}(s_i) \quad (1)$$

where the score $s_i$ denotes the score associated by the classifier $f$ to the item $a_1..a_n$ for being in the class $c_i$. Until now, we showed how the outputs of any classifier can be encoded as vectors of scores. Let us now see how to revise them to fit the user's requirements.

## *4.2 Strategies for revising a classifier's predictions*

Let us denote the set of objects to classify by $o_1,..,o_m$ with $o_i=(a_1a_2..a_n)_i$. Let us also denote the set of predictions made by the classifier $f$ by $v_1,..,v_m$ such that $f(o_i)=v_i$. Similarly, let us use $f_i$ (resp. $r_i$) to denote the class label predicted by $f$ (resp. the revision-based post-processor) for the object $o_i$. Assume also that we have a set of constraints $\mathscr{K}=\{K_1,..,K_w\}$ representing the extra domain knowledge and requirements to satisfy. In Section 2.1, we showed that any constraint $K_i \in \mathscr{K}$ can be expressed in the form $p_K(c_i)=\alpha_i \in [0,1]$. Then there are three situations to be considered:

1. **Case 1:** $\forall K_i \in \mathscr{K}$, $p_f(c_i)=\alpha_i$. This means that all the constraints $K_i$ (namely $p_K(c_i)=\alpha_i$) are already satisfied by the classifier $f$ (here, $p_f(c_i)$ denotes the proportion of items predicted in the class $c_i$ by the classifier $f$). Then the post-processor just predicts the same thing as the classifier, there is no relabeling of objects.
2. **Case 2:** $\exists K_i \in \mathscr{K}$, $p_f(c_i)>\alpha_i$. This situation happens when the classifier $f$ classifies more objects in a class $c_i$ than required by the domain knowledge. To satisfy the constraint $K_i$, some of the objects predicted as $c_i$ have to be relabeled in the other classes $c_k$ with $k \neq i$. The question that rises now is which items to relabel? This issue is dealt using selection criteria presented in the following section.
3. **Case 3:** $\exists K_i \in \mathscr{K}$, $p_f(c_i)<\alpha_i$. This situation happens if the classifier $f$ has not predicted enough objects in class $c_i$ meaning that some objects predicted by $f$ in the other classes $c_k$ with $k \neq i$ have to be revised and predicted by the post-processor in the class $c_i$. Here again, the question is which items from the other classes to relabel such that the constraint $K_i$ is satisfied? We provide selection criteria to deal with issue in the following section.

For *Case 2* and *Case 3*, many strategies can be adopted to select the objects to relabel while satisfying the set of constraints $\mathscr{K}$. The principles that our revision strategy follows are:

– *Minimize miss-classification cost:* This objective aims to minimize the overall miss-classification cost while satisfying the user's constraints. Such an objective requires i) relabeling only miss-classified items by the classifier and ii) relabel them in the right classes. In

order to achieve such an objective, we propose five criteria for selecting the items to relabel specifically designed to minimize the miss-classification cost.

– *Minimize relabelings:* This objective aims to ensure tractable computational complexity for the revision operation. Indeed, there are many solutions allowing to satisfy the set of input constraints. Our revision algorithm is designed to revise as few predictions as possible such that the user constraints are satisfied.

We use a heuristic algorithm to minimize the number of relabelings. It deals with the classifier predictions incrementally. It first satisfies the constraint $K_i$ requiring the largest items in class $c_i$, then it continues with the following constraints in a decrementing order. Note that it is enough to deal only with constraints of *Case 3* to satisfy the set of constraints $\mathscr{K}$. Moreover, in order to minimize relabelings, an item predicted in the class $c_i$ will not be relabeled if the corresponding constraint $K_i$ requires more items in $c_i$ than predicted by the classifier $f$. In Algorithm 1, the function *SelectItem*($\mathscr{O}$,$Cr$) allows

**Algorithm 1** Post-processing algorithm

**Input:** $\mathscr{O}$=$\{o_1,o_2,..,o_m\}$ // Objects to classify
$\mathscr{V}$ = $\{v_1,v_2,..,v_m\}$ // Score vectors output by $f$
$\mathscr{K}$ = $\{K_1,K_2,..,K_w\}$ // Constraints to satisfy
**Output:** $\mathscr{R}$=$\{r_1,r_2,..,r_m\}$ // Revised predictions
**procedure** POST-PROCESS($\mathscr{O},\mathscr{V},\mathscr{K}$)
  $\mathscr{R} \leftarrow \emptyset$
  $SK \leftarrow AscendingSort(\mathscr{K})$ //Get class order for relabeling
  **while** SK$\neq\emptyset$ **do**
    $c_j \leftarrow pop(SK)$ //Pick the highest class from $SK$
    $\mathscr{R}_j \leftarrow f_j$ //Set of object predicted in $c_j$ by $f$
    **while** $|R_j|<\alpha_j$ **do** //While constraint $K_j$ is not satisfied
      $o \leftarrow SelectItem(\mathscr{O},Cr)$ //Select object to relabel
      $r(o) \leftarrow c_j$ //Relabel $o$ in class $c_j$
      $\mathscr{R}_j \leftarrow \mathscr{R}_j \cup \{r(o)\}$
      $\mathscr{O} \leftarrow \mathscr{O} \setminus o$ //Discard $o$ from remaining items in $\mathscr{O}$
    **end while**
    $\mathscr{R} \leftarrow \mathscr{R} \cup \mathscr{R}_j$
  **end while**
  **return** $\mathscr{R}$
**end procedure**

to select an object to relabel among the remaining candidates in $\mathscr{O}$ using a criterion $Cr$ among the ones presented in the next section. It is clear that the complexity of this algorithm is polynomial in the number objects to post-process thanks to the incremental processing of items while satisfying the constraints of $\mathscr{K}$. Indeed, the costly operations are the sort function *AscendingSort*() (which is in the worst case in $O(k^2)$ using a quick sort algorithm with $k$ denoting the number of classes) and the complexity of the two nested *while* loops is less than $O(m^2)$ with $m$ denoting the number objects to post-process. In fact, each iteration discards an object and the function *SelectItem*() implementing our criteria is linear in the number of candidates $|\mathscr{O}|$.

## 5 Criteria for post-processing the predictions

In case a given constraint $\mathscr{K}_i$ is not satisfied (for instance, the proportion of items predicted by the classifier $f$ in $c_i$ is less than required by the constraint $\mathscr{K}_i$) then we need to relabel some items predicted by $f$ in the other classes and predict them in the target class $c_i$. There are many methods to select the items to relabel. In the following, we propose five criteria aiming at minimizing miss-classifications. The first three criteria are originally proposed in [6] in a computer security application.

### *5.1 MCTC (Maximize Confidence in the Target Class)*

This criterion interprets the scores $v_i$=$(s_1,s_2,..,s_k)_i$ associated with an object $o_i$ by the classifier $f$ as the confidence of $f$ that the right class of $o_i$ is $argmax((s_1,s_2,..,s_k)_i)$, namely the class corresponding to the highest score. Let $o_1$,..,$o_m$ be the set of objects that can be relabeled in $c_i$. The selected object $\acute{o}_j$ using the *MCTC* criterion is defined as follows:

$$\acute{o}_j = argmax_{j=1..m}(v[i]_j), \quad (2)$$

where $v[i]_j$ is the score $s_i$ of the target class $c_i$ in the vector $v_j$ of the scores associated by the classifier $f$ to $o_j$.

*Example 1.*Assume that we want to relabel one object among $o_1$, $o_2$ and $o_3$ in the class $c_4$.

| | $o_1$ | $o_2$ | $o_3$ |
|---|---|---|---|
| Classes | $v_1$ | $v_2$ | $v_3$ |
| $c_1$ | .1 | .4 | 0 |
| $c_2$ | .6 | .15 | .15 |
| $c_3$ | .2 | .3 | .45 |
| $c_4$ | .1 | .15 | **.4** |
| $f(o_i)$ | $c_2$ | $c_1$ | $c_3$ |

In this example, the objects $o_1$, $o_2$ and $o_3$ are predicted by the classifier $f$ respectively in $c_2$, $c_1$ and $c_3$. Since there is need to relabel one among them in the target class $c_4$, then the criterion *MCTC* selects to relabel the object $o_3$ as the score of $c_4$ in $v_3$ is the highest.

Intuitively, the *MCTC* criterion selects to relabel the object where the target class $c_i$ has the best score, it is the object where the confidence in $c_i$ is the highest that is selected. Doing so, we have more chances that the selected object by *MCTC* is in fact in the class $c_i$.

### *5.2 MCPC (Minimize Confidence in the Predicted Class)*

Here also the selection criterion interprets the scores $v_i$=$(s_1,s_2,..,s_k)_i$ associated with an object $o_i$ as the confidence of $f$ that the right class of $o_i$ is $argmax((s_1,s_2,..,s_k)_i)$. Since the objects to select among $o_1$,..,$o_m$ are considered as miss-classified, then another

way to select the object to relabel is to select the one classified with the lowest confidence. More formally,

$$\hat{o_j} = argmin_{j=1..m}(\max((s_1,..,s_k)_j)), \quad (3)$$

where $\max((s_1,..,s_k)_j)$ denotes the highest score among the ones associated by the classifier $f$ to the object $o_j$.

*Example 2.*Let the set of objects to relabel be $o_1$, $o_2$ and $o_3$ and let the target class be $c_4$.

| | $o_1$ | $o_2$ | $o_3$ |
|---|---|---|---|
| Classes | $v_1$ | $v_2$ | $v_3$ |
| $c_1$ | .1 | **.4** | 0 |
| $c_2$ | **.6** | .15 | .15 |
| $c_3$ | .2 | .3 | **.45** |
| $c_4$ | .1 | .15 | .4 |
| $f(o_i)$ | $c_2$ | $c_1$ | $c_3$ |

In this example, the criterion *MCPC* selects to relabel the object $o_2$ as its predicted class is $c_1$ and it is predicted with the lowest confidence (namely, .4).

## 5.3 MPTCD (Minimize the Predicted-Target Class Confidence Difference)

This criterion is a combination of the criteria *MCTC* and *MCPC* and aims at minimizing the gap between the predicted class and the target one.

$$\hat{o_j} = argmin_{j=1..m}(\max((s_1,..,s_k)_j) - v[i]_j). \quad (4)$$

*Example 3.*Let $o_1$, $o_2$ and $o_3$ be the set of objects to relabel and let the target class be $c_4$.

| | $o_1$ | $o_2$ | $o_3$ |
|---|---|---|---|
| Classes | $v_1$ | $v_2$ | $v_3$ |
| $c_1$ | .1 | **.4** | 0 |
| $c_2$ | **.6** | .15 | .15 |
| $c_3$ | .2 | .3 | **.45** |
| $c_4$ | *.1* | *.15* | *.4* |
| $f(o_i)$ | $c_2$ | $c_1$ | $c_3$ |

*MPTCD* will select to relabel the object $o_3$ since the gap between the score of the predicted class $c_3$ and the target one $c_4$ is .05 which is the smallest gap. The gap is interpreted here as a kind of confidence.

## 5.4 ME (Maximize the Entropy)

This criterion aims to select among the objects to relabel the one where the classifier $f$ is less confident in terms of entropy. This measure allows to assess the amount of uncertainty in a probability distribution. The entropy is maximal in case of uniform distributions and it is minimal if there is a value with all the probability mass (namely, 1) while all the other values have a zero probability. Intuitively, this criterion allows to relabel the object where the classifier $f$ is most uncertain (namely, less confident).

$$\hat{o_j} = argmin_{j=1..m}(entropy(s_1,..,s_k)_j), \quad (5)$$

where $entropy((s_1,..,s_k)_j)$=-$\sum_{i=1}^{k} s_i * log(s_i)$. The entropy-based criterion *ME* aims to relabel the objects where the classifier is most uncertain as it generally happens in case of novel and outlier objects.

*Example 4.*Assume that we have to choose among $o_1$, $o_2$ and $o_3$ an object to be relabeled in the target class $c_4$.

| | $o_1$ | $o_2$ | $o_3$ |
|---|---|---|---|
| Classes | $v_1$ | $v_2$ | $v_3$ |
| $c_1$ | .1 | .4 | 0 |
| $c_2$ | .6 | .15 | .15 |
| $c_3$ | .2 | .3 | .45 |
| $c_4$ | .1 | .15 | .4 |
| $f(o_i)$ | $c_2$ | $c_1$ | $c_3$ |
| $entropy(v_i)$ | 1.57 | 1.87 | 1.46 |

In this example, the criterion *ME* selects to relabel the object $o_2$ since the score vector $v_2$ contains the greatest entropy (uncertainty).

## 5.5 MMCC (Minimize Miss-Classification Cost)

This criterion allows to take into account both the scores output by the classifier $f$ and the miss-classification costs. Indeed, in some applications the cost of miss-classifying an object and predicting it in a wrong class $c_k$ is more costly than miss-classifying it and predicting it in another wrong class $c_j$. For instance, the cost of detecting an attack while there is no attack has not the same cost as not detecting any attack while there is actually one (see [31] for cost-sensitive classification problems). The choice of an object using the classifier's confidence and the miss-classification costs can be done as follows:

$$\hat{o_j} = argmin_{j=1..m}(\sum_{h=1}^{k} s_h * cost(f(o_j), c_i)), \quad (6)$$

where $cost(f(o_j), c_i)$ is the cost of miss-classification of $c_i$ in the class predicted by the classifier $f(o_j)$.

*Example 5.*In order to illustrate the *MMCC* criterion, let us assume that we deal with a cost-sensitive classification problem where the miss-classification costs are given in the following cost matrix.

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ |
|---|---|---|---|---|
| $c_1$ | 0 | 10 | 2 | 12 |
| $c_2$ | 3 | 0 | 2 | 10 |
| $c_3$ | 1 | 4 | 0 | 1 |
| $c_4$ | 5 | 10 | 1 | 0 |

Assume that we have to choose among $o_1$, $o_2$ and $o_3$ an object to relabel in the target class $c_4$.

| | $o_1$ | $o_2$ | $o_3$ |
|---|---|---|---|
| Classes | $v_1$ | $v_2$ | $v_3$ |
| $c_1$ | .1 | .4 | 0 |
| $c_2$ | .6 | .15 | .15 |
| $c_3$ | .2 | .3 | .45 |
| $c_4$ | .1 | .15 | .4 |
| $f(o_i)$ | $c_2$ | $c_1$ | $c_3$ |
| $cost(v_i)$ | 7.2 | 6.3 | 1.5 |

In this example, the criterion *MMCC* selects to relabel the object $o_3$ since the mean cost of score vector $v_3$ is the lowest.

## 6 Experimental setup

### *6.1 Datasets*

Table 1 gives the details of the used datasets. All these datasets are publically available (from the well-known UCI repository[1] and the KEEL imbalanced dataset repository[2]). Note that we selected different types of datasets with different characteristics. We selected multi-dimensional datasets with different sizes (# instances), dimensions (# attributes) and classification type problem (# classes). In particular, we selected some imbalanced[3] datasets especially because most classifiers are unable to achieve good classification rates on rare classes [12]. In Table 1, the datasets *contraceptive*, *balance*, *glass*, *yeast*, *lymphography*, *ecoli* and *thyroid* are imbalanced. Note that in Table 1 the datasets *column 2c* and *column 3c* are part of the UCI *Vertebral Column* dataset while *AU* denotes the UCI *AutoUniv* dataset.

**Table 1:** Datasets used in the experimental evaluation

| Dataset | # instances | # attributes | # classes |
|---|---|---|---|
| spambase | 4601 | 57 | 2 |
| dbworld | 64 | 4702 | 2 |
| column 2c | 310 | 7 | 2 |
| column 3c | 310 | 7 | 3 |
| AU | 25000 | 46 | 6 |
| contraceptive | 1473 | 10 | 3 |
| balance | 625 | 5 | 3 |
| glass | 214 | 10 | 7 |
| yeast | 1484 | 8 | 10 |
| lymphography | 148 | 18 | 4 |
| ecoli | 336 | 8 | 8 |
| thyroid | 720 | 22 | 3 |

Other datasets from the MDP repository[4] are also used to evaluate our approach on binary and imbalanced classification problems.

[1] https://archive.ics.uci.edu/ml/datasets.html
[2] http://sci2s.ugr.es/keel/imbalanced.php
[3] A dataset is said imbalanced if some classes are under represented.
[4] http://nasa-softwaredefectdatasets.wikispaces.com/

**Table 2:** MDP (NASA Metrics Data Program) datasets used in the experimental evaluation

| Dataset | # instances | # attributes | # classes |
|---|---|---|---|
| cm1 | 344 | 38 | 2 |
| jm1 | 9593 | 22 | 2 |
| kc1 | 2096 | 22 | 2 |
| kc3 | 200 | 40 | 2 |
| mc1 | 9277 | 39 | 2 |
| mc2 | 127 | 40 | 2 |
| mw1 | 264 | 38 | 2 |
| pc1 | 759 | 38 | 2 |
| pc2 | 1585 | 37 | 2 |
| pc3 | 1125 | 38 | 2 |
| pc4 | 1399 | 38 | 2 |
| pc5 | 17001 | 39 | 2 |

The results of the evaluated classifiers on the MDP datasets of Table 2 are given in the appendix.

### *6.2 Evaluated classifiers*

In order to evaluate our post-processing approach, we carried out experiments on both probabilistic classifiers (namely outputting probability distributions) and non probabilistic ones (namely outputting only one single class).

**Table 3:** Classifier details tested in the experimental studies

| Abrev. | Name | Reference | Category |
|---|---|---|---|
| NB | Naive Bayes | [19] | Probabilistic |
| TAN | Tree Augmented Naive Bayes | [19] | Probabilistic |
| BNK2 | Bayesian Network built with the K2 algorithm | [26] | Probabilistic |
| C4.5 | Decision Tree C4.5 | [32] | Non probabilistic |
| *k*NN | *k* Nearest Neighbor | [1] | Non probabilistic |

We evaluated as probabilistic classifiers the Naive Bayes classifier *NB*, *TAN*[5] and *BNK2*[6] [19]. These classifiers output posterior probability distributions for each object to classify. Such outputs are directly used by our post-processor when relabeling objects. As for non probabilistic classifiers, we used a C4.5 decision tree [32] and a *k*NN classifier where only class labels are predicted. Note that it is possible to obtain somehow probability

[5] Tree Augmented Naive Bayes.
[6] Bayesian Network built with the K2 algorithm.

distributions from decision trees and *k*NNs but in this work, we use only class label predictions.

## 6.3 Domain knowledge

Since it is difficult to show the interest of our approach on a specific application domain with real constraints, we first chose to perform experimental studies on widely used benchmarks for evaluating classifiers. In our experiments, we used datasets to build and evaluate the classifiers on them. As domain knowledge, we use different kinds of knowledge obtained only from training datasets.

– *Training Dataset Distribution (TDD):* Here, we use as domain knowledge the frequencies of the different classes in the training dataset. Let $\mathscr{D}$ be the training dataset and let $p_D(c_i)$ denote the frequency of items labeled as $c_i$ in $\mathscr{D}$. The knowledge we exploit here is $\mathscr{K}=\{K_1,..,K_k\}$ such that each constraint $K_i$ requires that $p_K(c_i)=p_D(c_i)$, namely the amount of objects labeled in $c_i$ after the post-processing step should be equal to the amount of objects in the class $c_i$ in the training dataset. As it will be shown in the obtained results, revising only with this available information allows to improve the classification rate using most classifiers.
– *Miss-Classification Rates (MCR):* The domain knowledge we exploit here is relative to the miss-classification rates (namely, miss-classification rates over all or some of the classes). Such rates are obtained by evaluating the classifier on the training dataset. It is easy to encode these rates as constraints composing $\mathscr{K}$ and exploit them for post-processing predictions. Here again, the used knowledge is available and it allows to improve the classifiers performances on most datasets.

## 6.4 Post-processing with knowledge on training datasets

### 6.4.1 Evaluation of probabilistic classifiers

Table 4 provides the results of the Naive Bayes *NB* classifier on the datasets of Table 1.

The experiments of Table 4 are done by revising the predictions of the *NB* classifier with the distributions (TDD) of the training datasets and the miss-classification rate of only one class of *NB* on the training datasets. The first six result columns of Table 4 denote respectively the PCC (Percentage of Correct Classification: It represents the proportion of correctly classified instances among all the classified instances) obtained with the *NB* classifier without any post-processing (column *NB*) while the remaining columns denote the results of post-processing the *NB* predictions using the criterion in the header of each column. In each cell, we give the results of revising with TDD knowledge and the results of revising with MCR knowledge between brackets. Note that for the *MMCC* criterion, we provide results obtained using a cost-matrix generated randomly. We provide results in terms of PCC, average classification cost without post-processing (in the last column) and average classification cost after post-processing (between brackets). For imbalanced datasets, the costs of miss-classifying rare classes are more important than miss-classifying majority ones. Finally, the results are obtained through a 10-fold cross-validation on the training datasets. The results of Table 4 show three main trends:

**Table 4:** Results of *NB* classifier evaluation on the datasets of Table 1.

| Dataset | NB | MCTC | MCPC | MPTCD | ME | MMCC | MMCC Cost |
|---|---|---|---|---|---|---|---|
| spam-base | 79.22% | 79.33% (80.42%) | 78.77% (79,46%) | 77.80% (81.83%) | 76.61% (77.63%) | 76% (82.13%) | 0.207 (0.178) |
| dbworld | 89.06% | 84.37% (87.50%) | 87.50% (90.63%) | 96.68% (96.87%) | 85.94% (90.62%) | 90.62% (90.62%) | 0.109 (0.093) |
| column 2c | 77.74% | 49.67% (65.80%) | 49.67% (68.06%) | 73.22% (80.96%) | 73.22% (79.67%) | 80.64% (80.96%) | 0.226 (0.190) |
| column 3c | 83.22% | 48.70% (61.61%) | 48.06% (79.03%) | 83.54% (83.54%) | 82.90% (83.22%) | 80.64% (83.54%) | 0.168 (0.165) |
| AU | 52% | 46.35% (48.62%) | 47.69% (52.36%) | 54.55% (54.81%) | 54.46% (54.76%) | 52.78% (54.81%) | 0.48 (0.452) |
| contra-ceptive | 49.69% | 50.98% (51.45%) | 50.03% (51.66%) | 55.24% (53.68%) | 51.66% (50.91%) | 51.32% (53.68%) | 0.503 (0.463) |
| balance | 90.08% | 64.48% (70.24%) | 64.96% (71.68%) | 89.76% (90.08%) | 89.44% (90.08%) | 89.76% (90.40%) | 0.099 (0.09) |
| glass | 47.19% | 46.73% (47.66%) | 44.39% (45.79%) | 47.19% (48.59%) | 50% (50.46%) | 53.74% (54.20%) | 0.528 (0.458) |
| yeast | 57.61% | 53.50% (57.88%) | 54.04% (58.15%) | 57.47% (58.69%) | 57.07% (57.74%) | 58.35% (58.55%) | 0.424 (0.415) |
| lympho-graphy | 83.78% | 77.02% (82.34%) | 77.02% (83.78%) | 83.10% (84.45%) | 81.75% (83.78%) | 82.43% (84.45%) | 0.162 (0.155) |
| ecoli | 85.41% | 77.67% (80.35%) | 77.38% (81.25%) | 83.92% (86.30%) | 80.95% (85.41%) | 85.11% (86.90%) | 0.146 (0.131) |
| thyroid | 95% | 94.44% (94.72%) | 96.52% (96.66%) | 94.86% (95.83%) | 96.38% (95.41%) | 95.41% (96.94%) | 0.05 (0.031) |

– The first trend is that on most the datasets using the MCR knowledge performs better than the classifier alone and better than the classifier with the post-processor exploiting the TDD knowledge. Indeed, revising with the MCR knowledge outperforms post-processing with the TDD knowledge with a gain in the PCC reaching sometimes 19% (see the results of *NB* classifier on the *column 2c* dataset in Table 1). Regarding the *MMCC* criterion, the revision decreases the miss-classification cost significantly meaning that the relabelings succeed in revising the labels of miss-classified instances of classes with high costs.
– The second trend is that on most the datasets the criteria *ME* and *MMCC* perform better than the *MCTC*, *MCPC* and *MPTCD* both when using TDD knowledge or the MCR knowledge. This result shows that when revising, the *ME* and *MMCC* criteria are better for selecting items to relabel among those miss-classified by the *NB* classifier. For the *ME* criterion, this is generally the case for outliers and

items containing some novelty (for instance, novel values in some attributes). Indeed, the *NB* classifier and more generally probabilistic ones, will return uniform posterior distributions making these items favorite candidates for the relabeling as their entropy is maximal. As for the *MMCC* criterion, it gives priority to relabel items with low posterior probability but taking into account the miss-classification cost.

–The improvements made by post-processing are more significant on the datasets where the *NB* alone has not good classification rate as on the *AU, contraceptive, glass* and *yeast* datasets. This result is somehow natural since it is hard for the post-processor to perform better than the base classifier if this latter already performs well on a dataset.

Table 5 provides the results of the *TAN* (Tree-Augmented Naive Bayes) classifier on the datasets of Table 1.

**Table 5:** Results of *TAN* classifier evaluation on the datasets of Table 1.

| Dataset | TAN | MCTC | MCPC | MPTCD | ME | MMCC | MMCC Cost |
|---|---|---|---|---|---|---|---|
| spam-base | 93.08% | 93.04% (93.15%) | 93.04% (91.78%) | 92.78% (93.35%) | 92.78% (93.35%) | 93.26% (93.84%) | 0.0692 (0.062) |
| dbworld | 79.68% | 78.12% (81.25%) | 78.12% (73.43%) | 78.12% (82.81%) | 78.12% (82.81%) | 81.25% (82.81%) | 0.2032 (0.172) |
| column 2c | 80.64% | 80.64% (80.96%) | 80.64% (80.96%) | 81.93% (82.25%) | 81.93% (82.25%) | 81.29% (81.93%) | 0.1936 (0.181) |
| column 3c | 78.06% | 76.77% (78.7%) | 76.12% (77.09%) | 77.41% (79.03%) | 76.77% (79.67%) | 80% (80.64%) | 0.2194 (0.194) |
| AU | 59.71% | 59.15% (59.77%) | 49.77% (59.39%) | 59.74% (60.18%) | 59.16% (59.72%) | 61.28% (61.6%) | 0.4029 (0.384) |
| contra-ceptive | 51.66% | 51.45% (51.73%) | 50.98% (51.32%) | 50.57% (52.61%) | 51.39% (52.41%) | 51.86% (52.47%) | 0.4834 (0.475) |
| balance | 71.68% | 65.92% (72%) | 64.96% (71.04%) | 72.32% (72.48%) | 72.48% (72.64%) | 72.64% (73.44%) | 0.2832 (0.266) |
| glass | 75.23% | 71.02% (75.23%) | 60.74% (74.76%) | 71.49% (75.7%) | 72.42% (76.17%) | 76.17% (76.63%) | 0.2477 (0.234) |
| yeast | 58.15% | 57.47% (58.22%) | 55.86% (58.01%) | 57.95% (58.49%) | 57.34% (58.36%) | 57.68% (58.55%) | 0.4185 (0.415) |
| lympho-graphy | 87.16% | 87.16% (87.16%) | 85.81% (86.48%) | 86.48% (87.16%) | 86.48% (87.16%) | 87.16% (87.83%) | 0.1284 (0.122) |
| ecoli | 80.05% | 80.35% (80.65%) | 80.05% (80.35%) | 81.25% (81.54%) | 80.95% (80.65%) | 80.35% (80.95%) | 0.1995 (0.191) |
| thyroid | 96.8% | 96.8% (97.08%) | 96.52% (96.52%) | 96.8% (97.08%) | 96.66% (97.36%) | 96.94% (97.5%) | 0.032 (0.025) |

The results of the Table 5 are similar to those of Table 4. However, given that *TAN* classifier is already more effective than *NB* on most datasets and as it is better before the use of post-processing then the post-processing results are not as significant as the improvements obtained in Table 4. Regarding the different revision criteria and the kind of knowledge we revise with, the results of the Table 5 allow us to draw the same conclusions as the trends drawn from Table 4.

In Table 6, we provide the results of the Bayesian network classifier *BNK*2 learnt using the K2 [14] algorithm. This evaluation is done on the datasets of Table 1.

From the results of Tables 6, one can notice that the results of the probabilistic classifier *BNK*2 share the main trends and conclusions of the results of *NB* and *TAN* classifiers of Table 4 and 5.

**Table 6:** Results of *BNK*2 classifier evaluation on the datasets of Table 1.

| Dataset | BNK2 | MCTC | MCPC | MPTCD | ME | MMCC | MMCC Cost |
|---|---|---|---|---|---|---|---|
| spam-base | 89.8% | 89.74% (89.91%) | 86.91% (89.74%) | 89.74% (89.95%) | 89.74% (89.95%) | 89.93% (89.98%) | 0.102 (0.100) |
| dbworld | 90.62% | 85.93% (89.06%) | 84.37% (87.5%) | 87.5% (92.18%) | 87.5% (92.18%) | 90.62% (92.18%) | 0.0938 (0.078) |
| column 2c | 76.45% | 80.64% (81.29%) | 70.32% (72.25%) | 80.96% (81.61%) | 80.96% (81.61%) | 81.29% (82.9%) | 0.2355 (0.171) |
| column 3c | 74.83% | 70.32% (73.22%) | 70.64% (72.25%) | 75.8% (76.45%) | 76.12% (75.8%) | 76.12% (76.77%) | 0.2517 (0.232) |
| AU | 54.92% | 37.4% (54.26%) | 37.11% (53.66%) | 54.22% (54.97%) | 53.56% (54.98%) | 54.94% (55.13%) | 0.4508 (0.449) |
| contra-ceptive | 51.12% | 50.91% (51.32%) | 50.78% (50.23%) | 50.91% (52.07%) | 50.71% (51.66%) | 51.05% (51.45%) | 0.4888 (0.485) |
| balance | 71.68% | 67.52% (72%) | 68.16% (71.84%) | 71.36% (73.12%) | 71.36% (73.12%) | 73.28% (75.2%) | 0.2832 (0.248) |
| glass | 73.36% | 66.35% (73.83%) | 59.81% (71.49%) | 71.02% (74.76%) | 70.56% (73.83%) | 71.96% (74.29%) | 0.2664 (0.257) |
| yeast | 56.73% | 56.33% (56.8%) | 53.36% (55.25%) | 56.46% (56.8%) | 56.19% (56.53%) | 56.87% (57.07%) | 0.4327 (0.429) |
| lympho-graphy | 85.81% | 85.13% (85.13%) | 83.1% (85.13%) | 87.16% (86.48%) | 86.48% (86.48%) | 85.81% (86.48%) | 0.1419 (0.135) |
| ecoli | 81.25% | 78.86% (81.54%) | 75.89% (80.35%) | 81.54% (82.44%) | 81.84% (82.44%) | 82.73% (83.33%) | 0.1875 (0.166) |
| thyroid | 96.66% | 96.52% (96.66%) | 96.38% (96.66%) | 96.38% (97.36%) | 96.66% (97.08%) | 97.22% (97.91%) | 0.0334 (0.021) |

### 6.4.2 Evaluation of non probabilistic classifiers

We use in the following experiments a standard C4.5 decision tree classifier [32] and a *k*NN classifier [1]. As for domain knowledge, we use only the class labels predicted by these classifiers. Table 7 (resp. Table 8) gives the results of C4.5 (resp. *k*NN) on the datasets of Table 1.

**Table 7:** Results of the *C*4.5 classifier evaluation on the datasets of Table 1.

| Dataset | C4.5 | MCTC | MCPC | MPTCD | ME | MMCC | MMCC Cost |
|---|---|---|---|---|---|---|---|
| spam-base | 92.97% | 92.91% (92.97%) | 92.97% (92.78%) | 92.91% (93%) | 92.91% (93.04%) | 93.28% (93.52%) | 0.0703 (0.065) |
| dbworld | 71.87% | 62.5% (70.31%) | 64.06% (70.31%) | 65.63% (71.87%) | 62.5% (71.87%) | 84.37% (85.94%) | 0.2813 (0.141) |
| column 2c | 81.61% | 80.64% (83.22%) | 78.06% (78.7%) | 82.58% (83.22%) | 82.58% (83.22%) | 83.22% (84.84%) | 0.1839 (0.152) |
| column 3c | 81.61% | 75.48% (81.93%) | 49.67% (80.96%) | 76.77% (81.29%) | 71.61% (80.64%) | 82.90% (83.23%) | 0.1839 (0.168) |
| AU | 64.25% | 64.08% (64.32%) | 63.96% (64.18%) | 64.35% (65.3%) | 64.37% (65.58%) | 64.1% (65.22%) | 0.3575 (0.348) |
| contra-ceptive | 53.76% | 52.95% (53.83%) | 52.81% (54.03%) | 54.31% (54.37%) | 53.9% (54.1%) | 54.79% (55.4%) | 0.4624 (0.446) |
| balance | 78.56% | 75.68% (77.44%) | 75.52% (77.44%) | 76.8% (80.64%) | 76.32% (79.52%) | 80.96% (81.92%) | 0.2144 (0.181) |
| glass | 67.75% | 64.95% (67.28%) | 66.35% (67.75%) | 67.75% (69.16%) | 64.48% (68.69%) | 69.16% (70.56%) | 0.3225 (0.294) |
| yeast | 55.86% | 55.86% (55.99%) | 55.72% (56.13%) | 55.79% (55.99%) | 55.86% (56.06%) | 56.06% (56.74%) | 0.4414 (0.433) |
| lympho-graphy | 80.4% | 80.4% (80.4%) | 80.4% (80.4%) | 80.08% (81.08%) | 80.4% (81.75%) | 80.41% (82.43%) | 0.196 (0.176) |
| ecoli | 84.22% | 84.22% (84.52%) | 83.03% (83.92%) | 84.22% (84.52%) | 83.33% (84.82%) | 84.52% (85.42%) | 0.1578 (0.146) |
| thyroid | 98.61% | 98.61% (98.75%) | 98.61% (98.47%) | 98.47% (98.75%) | 98.33% (98.61%) | 98.75% (98.89%) | 0.0139 (0.011) |

The *C*4.5 classifier is well-known and it is recognized among the most efficient ones in the literature. On the datasets of Table 1, one can notice that compared to the results of probabilistic classifiers *NB*, *TAN* and *BNK*2, *C*4.5 classifier provides better performances on most datasets. Regarding the post-processing, one can draw almost the same conclusions as those drawn from the results of the probabilistic classifiers, namely i) on most the datasets, the revision improves the PCC (for example, the PCC is improved by 14% on the *dbworld* dataset using the *MMCC* criterion), ii) revision with the MCR knowledge provides better results than post-processing based on TDD knowledge and iii) the *MMCC* criterion provides better results than the other criteria.

As for the results of the *k*NN classifier given in Table 8, the results are slightly mixed in comparison with the probabilistic classifiers *NB*, *TAN* and *BNK*2 and those of *C*4.5 decision tree. Indeed, on some datasets, there is a slight deterioration in the PCC while on others, there are slight improvements. But on the majority of datasets, the use of MCR knowledge in post-processing provides better results than when using the TDD knowledge and the *MMCC* criterion provides the best results.

**Table 8:** Results of the *k*NN classifier evaluation on the datasets of Table 1.

| Dataset | *k*NN | MCTC | MCPC | MPTCD | ME | MMCC | MMCC Cost |
|---|---|---|---|---|---|---|---|
| spam-base | 90.76% | 90.39% (90.39%) | 90.39% (90.39%) | 90.39% (91.13%) | 90.39% (91.11%) | 91.05% (91.59%) | 0.0924 (0.084) |
| dbworld | 79.68% | 78.12% (78.12%) | 67.18% (68.75%) | 81.25% (82.81%) | 79.68% (82.81%) | 81.25% (84.38%) | 0.2032 (0.156) |
| column 2c | 81.61% | 81.29% (82.25%) | 81.29% (81.93%) | 81.29% (82.58%) | 81.29% (82.58%) | 81.93% (82.9%) | 0.1839 (0.171) |
| column 3c | 78.38% | 77.74% (79.03%) | 74.51% (77.41%) | 77.74% (80%) | 78.06% (79.68%) | 79.35% (80.97%) | 0.2162 (0.190) |
| AU | 41.11% | 41.11% (41.13%) | 41.01% (41.2%) | 41.16% (41.43%) | 40.43% (41.13%) | 41.15% (41.42%) | 0.5889 (0.586) |
| contra-ceptive | 43.1% | 40.19% (42.97%) | 42.97.1% (43.1%) | 43.17% (44.73%) | 43.31% (43.85%) | 43.92% (44.12%) | 0.569 (0.559) |
| balance | 87.36% | 85.6% (87.2%) | 77.4% (86.72%) | 87.68% (89.62%) | 86.88% (87.52%) | 88.8% (90.24%) | 0.1264 (0.0978) |
| glass | 70.09% | 47.66% (50%) | 67.75% (70.09%) | 71.03% (71.96%) | 69.15% (70.56%) | 71.5% (72.9%) | 0.2991 (0.271) |
| yeast | 52.29% | 47.16% (49.79%) | 45.88% (51.81%) | 52.22% (52.49%) | 51.88% (52.29%) | 52.63% (52.7%) | 0.4771 (0.473) |
| lympho-graphy | 80.4% | 79.05% (80.4%) | 77.7% (79.72%) | 80.4% (81.76%) | 80.4% (81.08%) | 81.76% (83.78%) | 0.196 (0.162) |
| ecoli | 80.35% | 66.66% (78.27%) | 77.08% (80.35%) | 80.06% (81.25%) | 79.46% (80.35%) | 79.76% (82.14%) | 0.1965 (0.178) |
| thyroid | 90% | 87.77% (90.41%) | 87.5% (89.86%) | 90.13% (92.63%) | 89.86% (91.94%) | 91.25% (93.19%) | 0.1 (0.068) |

The conclusions that can be drawn from the results of Table 7 and Table 8 on non probabilistic classifiers are basically the same as the three trends characterizing the evaluation of Tables 4, 5 and 6.

## 7 Case study: Exploiting domain knowledge in computer security

This section provides a realistic application needing to post-process the predictions of classifiers and detection/prediction systems with domain knowledge and user's constraints in the computer security field.

### *7.1 Intrusion detection and alert correlation*

The objective of computer security is to protect the system against any attempt to violate the security policy. Two kinds of solutions are generally used to ensure the confidentiality, integrity and availability of information and services of an information system:

– *Prevention solutions:* Like fire-walls, ciphering technologies, access control, etc., such solutions aim to prevent the violation of the security policy.
– *Detection solutions:* Because there is no guarantee that the used prevention solutions provide a complete security, there is need to use tools to detect the intrusions and attacks that overcome the prevention security tools. Examples of detection tools are intrusion detection systems (IDSs for short)[3] such as Snort IDS[7], alert correlation [28][18] and activity monitoring [27][23].

Intrusion detection consists in analyzing the activities (ex. network traffic, log files, etc.) to detect in real-time or offline any attempt to violate the security policy. IDSs act as burglar alarms and they are either misuse-based [33] or anomaly-based [30] or a combination of both the approaches in order to exploit their mutual complementarities [37].

Computer security practitioners often deploy multiple security products and solutions in order to increase the detection rates by exploiting their mutual complementarities. For instance, misuse-based IDSs are often combined with anomaly-based ones in order to detect both old and novel attacks and anomalies. It is important to note that all exiting anomaly-based approaches have a major drawback consisting in very high false alarm rates. These systems build profiles and models of legitimate activities and detect attacks by computing the deviations of the analyzed activities from normal activity profiles. In the literature, most anomaly-based IDSs are novelty or outlier approaches [30][35] adapted for the intrusion detection problem. Moreover, all modern IDSs (even the *de facto* network Snort IDS are well-known to trigger large amounts of alerts most of which are redundant and false ones. This problem is due to several reasons such as bad parameter settings and inappropriate IDS tuning, etc. [36]. As a consequence, huge amounts of alerts are daily reported

[7] http://snort.org/

making the task of the security administrators time-consuming and inefficient. In order to cope with such quantities of alerts, alert correlation approaches are used [17][16].

Alert correlation [17][16] consists in analyzing the alerts triggered by one or multiple IDSs and other security tools in order to provide a *synthetic* and *high-level* view of the *interesting* malicious events targeting the information system. The input data for alert correlation tools is gathered from various sources such as IDSs, fire-walls, web server logs, etc. Correlating alerts reported by multiple analyzers and sources has several advantages such as exploiting the complementarities of multiple analyzers. The main objectives of alert correlation are:

1. *Alert reduction and Redundant alerts elimination:* The objective of alert correlation here is to eliminate redundant alerts by aggregating or fusing similar alerts [17]. In fact, IDSs often trigger large amounts of redundant alerts due to the multiplicity of IDSs and the repetitiveness of some malicious events such scans, floodings, etc.
2. *Multi-step attack detection:* Most IDSs report only elementary malicious events while several attacks perform through multiple steps where each step can be reported by an alert. Detecting multi-step attacks requires analyzing the relationships and connections between several alerts [7][11][29].
3. *Alert filtering and prioritization:* Among the huge amount of triggered alerts, security administrators must select a subset of alerts according to their dangerousness and the contexts. Alerts filtering/prioritization aims at presenting to the administrators only the alerts they want to analyze[8].

In the literature, alert correlation approaches are often grouped into similarity-based approaches [17], predefined attack scenarios [29], pre and post-conditions of individual attacks [16] and statistical approaches [40][22].

We illustrate here the need to post-processing prediction or detection systems outputs in order to fit the user knowledge and requirements.

– **In intrusion detection**, a typical domain knowledge requiring to revise the predictions and decisions of a detection system (here an IDS for instance) is that security operators may know for example that there can not be successful attacks against a the Web server given that there is no Web server in the monitored network. Here the domain knowledge requires to relabel for example the alerts saying that a Web attack is currently undergoing as normal activity for instance. This same scenario is also encountered for attacks targeting systems and versions that are not present or running in the network. In this case, it is knowledge about application domain that allows to revise the predictions of the IDS.

– **In alert correlation**, the security operators are faced continuously to very large volumes of alerts generated by the IDSs. Since it is impossible to analyze and manually check all the triggered alerts, security operators often prefer to select a subset of alerts according to operators' availability and the dangerousness of the attacks. Typically, this is a post-processing task of the predictions IDSs to fit the constraints of the security operators. In this case, the security operator may want to select only the 1000 most dangerous or most likely attacks among all those generated by the IDSs. In this case, the knowledge requiring to post-process the predictions of the detection systems is the constraints and preferences of the security operators. This task is similar in some sense to prioritizing alerts (ranking the triggered alerts according to user specified criteria [9]).

In the above scenarios, it is clear that it really makes sense to post-process the predictions of security tools which can be non probabilistic (outputting only symbolic information like alerts that can be though of as classes) or probabilistic (for instance, when the detection is based on a probabilistic model as in SPADE system[8]). In order to evaluate our post-processing approach, we carried out experimentations similar to the ones of the experimental study Section but they are done on real and representative data from intrusion detection in Web attacks and alert correlation.

## *7.2 Experiments on the intrusion detection problem*

In order to evaluate our post-processing approach on real dataset, we use the Webtraffic dataset of Table 9. It contains real network traffic data collected on a university campus. 18 days of network traffic were collected and the volume of collected data is 100 Giga bytes. The raw data is preprocessed into connection records described by relevant features as described in [10]. For our experimentations, only 15 Giga bytes of traffic were used. The normal traffic is real and includes inbound and outbound *http* connections captured with TCPDump sniffer. While the attacks are simulated and reproduced from [21] which are among the richest and well documented Web attacks databases (several source codes and scripts of these attacks are publicly available[9]). These attacks have various strategies and objectives:

– *Denial of Service (DOS):* submerge the server with requests (attacks by flooding) or provoke the services stop or deceleration (attacks by buffer-overflow, errors of decoding and interpretation of data, bad allowance, etc.)

[8] http://sourceforge.net/projects/snortspade/
[9] http://www.i-pi.com/HTTP-attacks-JoCN-2006/

–*Scans of vulnerabilities (SOV):* search possible breaches and errors of con- figuration in the Web applications in order to exploit them for malevolent ends.
–*Unauthorized Accesses (UA):* reach the system or reveal information with unauthorized ways. The unauthorized access to the system is carried out with command injections, Shell codes, etc. The unauthorized access to the data can be done with SQL injection, Cross Scripting Site, Directory Climbing, etc.

**Table 9:** Webtraffic dataset

| | Training dataset | | Test dataset | |
|---|---|---|---|---|
| **Class** | **#** | **%** | **#** | **%** |
| Normal connections | 55911 | 56.29% | 61378 | 88.875% |
| *Vulnerability scans* | *30979* | *31.19%* | *4456* | *6.45%* |
| Flooding | 12375 | 12.46% | 3159 | 4.57% |
| *Buffer overflow* | *9* | *0.009%* | *15* | *0.022%* |
| URL decoding error | 36 | 0.036% | 21 | 0.03% |
| Directory Climbing | 4 | 0.004% | 1 | 0.001% |
| Values misinterpretation | 2 | 0.002% | 2 | 0.003% |
| **Cross Site Scripting** | **0** | **0%** | **6** | **0.009%** |
| **SQL injection** | **0** | **0%** | **14** | **0.02%** |
| **Command injection** | **0** | **0%** | **9** | **0.013%** |
| Total | **99316** | **100%** | **69061** | **100%** |

Note that the testing dataset involves new attacks which do not appear in training dataset. Indeed, in order to evaluate the efficiency to detect new attacks which is a serious issue in IDSs, the testing data set of Webtraffic includes normal real *http* connections as well as known attacks and new ones (attacks in bold in Table 9).

Table 10 provides the results of probabilistic classifiers *NB*, *TAN* and *BNK*2 and non probabilistic classifier *C*4.5 decision tree on the dataset of Table 9.

**Table 10:** Results of the classifiers on the Webtraffic dataset

| Classifier | Without Revision | MCTC | MCPC | MPTCD | ME | MMCC | MMCC Cost |
|---|---|---|---|---|---|---|---|
| NB ALL | 93.89% | 97.94% (98.06%) | 89.25% (97.95%) | 97.91% (98.35%) | 97.77% (98.18%) | 97.98% (98.18%) | 0.061 (0.018) |
| TAN ALL | 97.43% | 98.05% (98.7%) | 96.16% (98.02%) | 98.51% (98.66%) | 97.82% (98.58%) | 98.75% (99.28%) | 0.026 (0.007) |
| BNK2 ALL | 94.5% | 94.72% (94.5%) | 90.26% (94.72%) | 97.83% (98.26%) | 94.69% (99.31%) | 97.29% (99.31%) | 0.055 (0.007) |
| C4.5 ALL | 93.33% | 93.32% (93.37%) | 87.04% (92.88%) | 93.33% (93.39%) | 93.33% (93.39%) | 93.35% (99.18%) | 0.067 (0.008) |
| NB 2classes | 97.81% | 95.69% (96.87%) | 95.69% (96.87%) | 96.87% (98.25%) | 96.87% (98.39%) | 97.84% (98.13%) | 0.022 (0.018) |
| TAN 2classes | 96.09% | 94.76% (97.51%) | 94.76% (97.51%) | 97.51% (98.25%) | 97.51% (98.25%) | 98.45% (98.7%) | 0.039 (0.013) |
| BNK2 2classes | 99.31% | 98.75% (99.46%) | 98.75% (99.46%) | 99.46% (99.32%) | 99.46% (99.48%) | 99.61% (99.93%) | 0.007 (0.001) |
| C4.5 2classes | 93.3% | 99.59% (99.79%) | 87.02% (93.1%) | 99.59% (99.8%) | 99.59% (99.8%) | 97.5% (98.5%) | 0.067 (0.015) |

In Table 10, we first provide (in the rows tagged *ALL*) the results of *NB*, *TAN*, *BNK*2 and *C*4.5 classifiers on the dataset of Table 10 and we provide the results of these classifiers where the dataset Webtraffic is modified and consider only two classes (*Normal* and *Attack*) in the rows tagged 2*classes*. The conclusions that can be drawn from the results of Table 10 are basically the same as those drawn form the results of the previous experimental evaluation Section. It particular, one can observe real improvements of the PCC achieved by all the revision criteria over the four evaluated classifiers. Note that in this section, we use only the MCR knowledge.

In order to have more insights into the results (namely, which classes are improved), Tables 11, 12, 13 and 14 provide the PCC rates on both the *Normal* and *Attack* classes of *NB*, *TAN*, *BNK*2 and *C*4.5 classifiers on the dataset of Table 10 .

**Table 11:** Result details of the *NB* classifier on Webtraffic datatset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 97.57% | 97.59% | 97.59% | 98.07% | 98.21% | 97.93% |
| Attack | 99.71% | 91.05% | 91.05% | 99.71% | 99.79% | 99.71% |
| PCC | 97.81% | 96.87% | 96.87% | 98.25% | 98.39% | 98.13% |

**Table 12:** Result details of the *TAN* classifier on the Webtraffic dataset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 99.94% | 99.41% | 99.41% | 99.29% | 99.29% | 99.63% |
| Attack | 65.13% | 82.21% | 82.21% | 89.89% | 89.89% | 91.2% |
| PCC | 96.09% | 97.51% | 97.51% | 98.25% | 98.25% | 98.7% |

**Table 13:** Result details of the *BNK*2 classifier on the Webtraffic dataset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 99.94% | 99.94% | 99.94% | 99.94% | 99.94% | 99.94% |
| Attack | 94.27% | 95.62% | 95.62% | 94.32% | 95.62% | 99.79% |
| PCC | 99.31% | 99.46% | 99.46% | 99.32% | 99.48% | 99.93% |

One can notice in the results of Tables 11, 12, 13 and 14 that there are significant improvements in the attack detection rate. Indeed, despite the fact that the Webtraffic dataset is imbalanced and contains several new attacks (which are not included in the training data), the attack detection rate has been significantly enhanced. For instance, in Table 14 the *C*4.5 classifier detected only 41.36% of the attacks while after the post-processing, 99.14% of the attacks were detected using *ME* criterion.

**Table 14:** Result details of the *C*4.5 decision tree on the Webtraffic dataset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 99.77% | 99.78% | 99.7% | 99.88% | 99.88% | 99.83% |
| Attack | 41.36% | 99.89% | 40.05% | 99.14% | 99.14% | 87.78% |
| PCC | 93.3% | 99.79% | 93.1% | 99.8% | 99.8% | 98.5% |

Regarding the detection of new attacks (namely those that do not appear in the training data set), the obtained results are some how similar to the results on other intrusion detection benchmarks using decisions trees and NB classifiers that can be found for instance in [4]. More precisely, the detection rates on most intrusion detection benchmarks are often high for known attacks and normal activities and very insufficient for novel and rare attacks. In our experiments, the detection rate of some new attacks has been improved but it is not equivalent to the improvements achieved on some known attacks. Indeed, since the new attacks are novel, their posterior probabilities are very low (in general, new attacks are too different from known attacks), hence the revision criteria do not recover them in this case.

## *7.3 Experiments on the alert correlation problem*

In this experiment, we used a dataset built on real alert log files produced by Snort IDS monitoring a university campus network. These alert logs represent three months activity. This data consists in alerts generated by Snort IDS gathered in IDMEF[10] format then preprocessed into CSV (Comma Separated Values). In these experiments, we use the datasets of Table 15 obtained from real IDMEF alerts reported during three months by the Snort IDS. We first preprocessed the first month of collected alerts in order to build the training data set and preprocessed the second month to build the testing set. Table 15 provides details on the attacks we used in our experimentations.

Among the attacks detected by Snort IDS, we selected 9 Web-based attacks to predict on the basis of the alerts that often precede/prepare these attacks. All these attacks are associated with a high severity level and are targeting either Web servers or related web-based applications. Such attacks may result in arbitrary code execution and full control of the targeted system. Interested readers can refer to Snort IDS signature database for additional information and references on these attacks.

The results of Table 16 allow to draw the same conclusion as the results of the *NB*, *TAN*, *BNK*2 and *C*4.5 classifiers on the Webtraffic dataset. Similarly, the result details given in Tables 17, 18, 19 and 20 allow also to draw the same conclusions except the fact that the only slight improvements are achieved on the attack rates.

[10] IDMEF stands for the Intrusion Detection Message Exchange Format. http://www.ietf.org/rfc/rfc4765.txt

**Table 15:** Training and testing datasets distributions of the alerts dataset

| | | Training set | | Testing set | |
|---|---|---|---|---|---|
| Sid | Snort alert name | # | % | # | % |
| 1091 | WEB-MISC ICQ Webfront HTTP DOS | 87 | 0,18% | 6 | 0,01% |
| 2002 | WEB-PHP remote include path | 50 | 0,10% | 231 | 0,47% |
| 2229 | WEB-PHP viewtopic.php access | 5169 | 10,42% | 1580 | 3,20% |
| 1012 | WEB-IIS fpcount attempt | 3 | 0,01% | 10 | 0,02% |
| 1256 | WEB-IIS CodeRed v2 root.exe access | 2 | 0,004% | 3 | 0,01% |
| 1497 | WEB-MISC cross site scripting attempt | 5602 | 11,30% | 7347 | 14,90% |
| 2436 | WEB-CLIENT Microsoft wmf metafile access | 145 | 0,29% | 53 | 0,11% |
| 1831 | WEB-MISC jigsaw dos attempt | 659 | 1,33% | 153 | 0,31% |
| 1054 | WEB-MISC weblogic/tomcat .jsp view source... | 3412 | 6,88 % | 3885 | 7,88% |

**Table 16:** Results of the classifiers on the alerts dataset

| Classifier | Without Revision | MCTC | MCPC | MPTCD | ME | MMCC | MMCC Cost |
|---|---|---|---|---|---|---|---|
| NB | 90.17% | 87.77% (90.24%) | 87.27% (90.2%) | 89.7% (90.36%) | 89.95% (90.37%) | 90.26% (90.41%) | 0.0983 (0.096) |
| TAN | 90.78% | 89.84% (90.87%) | 87.32% (90.46%) | 90.56% (90.88%) | 90.51% (90.79%) | 90.88% (90.92%) | 0.092 (0.091) |
| BNK2 | 90.5% | 87.97% (90.38%) | 87.89% (90.22%) | 90.18% (90.57%) | 90.19% (90.57%) | 90.73% (90.75%) | 0.095 (0.092) |
| C4.5 | 91.07% | 86.86% (91.17%) | 86.82% (91.14%) | 90.19% (91.2%) | 90.12% (91.12%) | 91.16% (91.2%) | 0.0893 (0.088) |
| kNN | 91.04% | 87.13% (91.1%) | 87.06% (90.93%) | 90.81% (91.15%) | 90.78% (91.04%) | 91.17% (91.28%) | 0.089 (0.087) |

**Table 17:** Result details of the *NB* classifier on the alerts dataset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 95.92% | 95.94% | 95.93% | 95.97% | 95.97% | 95.97% |
| Attack | 77.14% | 77.34% | 77.22% | 77.71% | 77.75% | 77.9% |
| PCC | 90.87% | 90.94% | 90.9% | 91.06% | 91.07% | 91.11% |

**Table 18:** Result details of the *TAN* classifier on the alerts dataset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 98.29% | 98.29% | 98.29% | 98.28% | 98.29% | 98.29% |
| Attack | 72.48% | 72.81% | 71.27% | 72.84% | 72.5% | 72.98% |
| PCC | 91.34% | 91.43% | 91.02% | 91.44% | 91.35% | 91.48% |

In Table 21, we provide the results of post-processing the non probabilistic C4.5 decision tree classifier predictions with different thresholds. In each column, we give the results of the selecting only a proportion $x$% of the predictions made by the C4.5 decision tree as attacks using the revision criteria presented in this paper.

**Table 19:** Result details of the *BNK*2 classifier on the alerts dataset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 96.78% | 96.78% | 96.72% | 96.72% | 96.72% | 96.74% |
| Attack | 75.42% | 74.97% | 74.54% | 75.85% | 75.85% | 76.45% |
| PCC | 91.03% | 90.91% | 90.75% | 91.1% | 91.1% | 91.28% |

**Table 20:** Result details of the *C*4.5 decision tree on the alerts dataset

| Class | Without revision | MCTC | MCPC | MPTCD | ME | MMCC |
|---|---|---|---|---|---|---|
| Normal | 98.57% | 98.59% | 98.59% | 98.59% | 98.58% | 98.59% |
| Attack | 73.03% | 73.32% | 73.21% | 73.42% | 73.17% | 73.42% |
| PCC | 91.69% | 91.79% | 91.76% | 91.82% | 91.74% | 91.82% |

**Table 21:** Result of the *C*4.5 decision tree on the alerts dataset with different thresholds.

| Threshold | 2% | 5% | 10% | 20% | 100% |
|---|---|---|---|---|---|
| MCTC | 43.59% | 77.38% | 88.69% | 94.34% | 98.16% |
| MCPC | 9.62% | 63.75% | 81.88% | 90.94% | 40.6% |
| MPTCD | 42.86% | 76.86% | 88.72% | 94.34% | 98.19% |
| ME | 42.86% | 76.86% | 88.72% | 94.34% | 98.19% |
| MMCC | 36.23% | 74.42% | 87.23% | 93.61% | 97.95% |

As it can be seen in Table 21, even when we just pick a small proportion of predictions detecting attacks by C4.5, the majority of selected items after post-processing are indeed attacks. For example, when we select just 10% of the predictions classified attacks by C4.5, all the revision criteria select over than 80% of attacks. Note that the same experiment is carried out using the Naive Bayes classifier and we obtained similar results.

## 8 Discussions and concluding remarks

This paper dealt with a novel and important issue in classification. More precisely, it addressed the problem of exploiting the available domain knowledge in order to achieve two objectives: i) improve the classifier efficiency and ii) fit the user requirements. This issue is addressed as a general problem and it can be encountered in many applications, typically where users have specific domain constraints that their detection/prediction models should satisfy.

In [2], the authors proposed a method for classifying data items with some uncertain observations using a possibilistic decision trees. Then, they proposed a method to evaluate the classifier taking into account the uncertainty of the predictions. Clearly, our work is complementary since we allow revising the outputs of a classifier to fit the user requirements. Indeed, using possibilistic decision trees as first-level classifiers fully make sense especially if the data objects to be classified can be uncertain. In [5] the authors also dealt with evaluation criteria for probabilistic classifiers but did not address the issue of revising the predictions. Their objective was to define alternative and more informative evaluation criteria in case where the prediction of the classifier is probability distribution over the set of classes instead of just only one class.

The selection criteria proposed in this paper are based on natural ideas and aim at minimizing miss-classifications while fitting all the considered domain knowledge. Note that the criteria *MCTC*, *MCPC* and *MPTCD* are originally proposed in [6] within a computer security application. In this paper, two more efficient criteria are proposed and the five criteria are evaluated on widely used benchmarks in the classification community. The obtained experimental results are appealing and very encouraging since **i) the proposed approach guarantees that the post-processed predictions fit well the domain knowledge constraints** and **ii) does not deteriorate the prediction system classification rates but may even improve it**.

Algorithm 1 proposed to post-process the classifier predictions gives priority to minimizing the number of relabelings to guarantee a better post-processing time complexity. This of course affects the miss-classification rates. A better compromise between complexity and minimizing miss-classifications requires to reconsider the predictions of items even those predicted in the target class (currently, the algorithm don't reconsider items predicted in the target class $c_i$ if the corresponding constraint $\mathscr{K}_i$ requires more items in $c_i$).

To sum up, the contributions of the paper are:

1. Proposing a unifying encoding for classifiers and prediction models outputs in general. It also provides a unifying encoding of different types of domain knowledge such as generic information about the objects to classify, user preferences and constraints.
2. Proposing a polynomial post-processing algorithm to revise the predictions of a classifier guaranteeing revised predictions in full agreement with the domain knowledge.
3. Different criteria are proposed to select among the items to relabel the ones that best allow to achieve the post-processing objectives.
4. Extensive experimental studies are carried out showing that the proposed post-processing approach can achieve significant improvements especially on datasets where the classifiers have poor efficiency as on imbalanced datasets.
5. A case study on two typical computer security problems is provided. In these problems it really makes sense to revise the predictions of a prediction/detection system with the users' domain

knowledge, constraints and preferences. In particular, we showed that exploiting some background knowledge allows to improve the attack detection rates.

It is important to point out that our approach is designed as a plug-in to be combined with any prediction model be it a probabilistic or non probabilistic classifier or even any detection or prediction model (such as spam filters, IDSs [3], etc.). In particular, this approach can be adapted for the classification with reject option [13] to make alternative predictions instead of just rejecting some items. Other future works will deal with this issue in regression problems, multiple classifier systems and consider this problem in real-time contexts.

## 9 Appendix

### *9.1 Evaluation on binary classification problems of datasets of Table 2*

In the following, we provide in Table 22 and Table 23 the results of evaluating the *NB* classifier and C4.5 decision tree classifier on the MDP datasets of Table 2.

**Table 22:** Results of the NB classifier evaluation on MDP datasets of Table 2

| Dataset | *NB* | MCTC | MCPC | MPTCD | ME | MMCC | Cost |
|---|---|---|---|---|---|---|---|
| CM1 | 82.26% | 83.72% (86.04%) | 74.41% (80.52%) | 82.55% (87.2%) | 82.55% (87.2%) | 82.26% (87.79%) | 0.1774 (0.122) |
| JM1 | 81.41% | 74.02% (81.59%) | 79.65% (81.21%) | 74.02% (81.5%) | 74.02% (81.5%) | 73.85% (81.13%) | 0.186 (0.188) |
| KC1 | 82.44% | 81.77% (82.91%) | 77.81% (81.77%) | 81.77% (83.68%) | 81.77% (83.68%) | 82.44% (84.3%) | 0.176 (0.157) |
| KC3 | 78.5% | 77% (82%) | 67% (75.5%) | 77% (83%) | 77% (83%) | 79% (84%) | 0.215 (0.16) |
| MC1 | 94.11% | 98.85% (98.87%) | 92.13% (93.16%) | 98.7% (98.92%) | 98.7% (98.92%) | 98.84% (98.98%) | 0.059 (0.010) |
| MC2 | 73.22% | 66.92% (74.44%) | 54.33% (70.68%) | 66.92% (74.01%) | 66.92% (74.01%) | 66.92% (73.22%) | 0.268 (0.268) |
| MW1 | 81.81% | 83.33% (87.12%) | 79.16% (78.4%) | 86.36% (87.12%) | 86.36% (87.12%) | 87.87% (89.01%) | 0.182 (0.109) |
| PC1 | 88.27% | 88.4% (89.32%) | 87.48% (87.08%) | 89.45% (91.17%) | 89.45% (91.17%) | 89.72% (91.43%) | 0.117 (0.085) |
| PC2 | 95.45% | 98.1% (98.67%) | 86.68% (82.96%) | 98.23% (98.8%) | 98.23% (98.8%) | 98.54% (98.8%) | 0.045 (0.012) |
| PC3 | 35.82% | 76% (77.24%) | 64.71% (54.04%) | 83.28% (82.75%) | 83.28% (82.75%) | 83.55% (85.33%) | 0.642 (0.146) |
| PC4 | 86.91% | 85.99% (86.34%) | 84.7% (84.84%) | 85.99% (87.77%) | 85.99% (87.77%) | 86.7% (87.84%) | 0.131 (0.122) |
| PC5 | 96.3% | 96.48% (96.76%) | 93.79% (95.18%) | 96.64% (97.01%) | 96.64% (97.01%) | 96.44% (96.96%) | 0.037 (0.030) |

**Table 23:** Results of the C4.5 classifier evaluation on MDP datasets of Table 2

| Dataset | C4.5 | MCTC | MCPC | MPTCD | ME | MMCC | Cost |
|---|---|---|---|---|---|---|---|
| CM1 | 85.46% | 83.13% (86.33%) | 83.13% (79.36%) | 83.13% (87.2%) | 83.13% (87.2%) | 85.46% (87.79%) | 0.1454 (0.122) |
| JM1 | 79.93% | 77.52% (81.48%) | 77.52% (79.13%) | 76.85% (80.51%) | 76.85% (80.52%) | 81.66% (81.74%) | 0.200 (0.182) |
| KC1 | 84.16% | 81.77% (84.25%) | 81.77% (83.68%) | 81.77% (84.49%) | 81.77% (84.49%) | 84.44% (84.73%) | 0.158 (0.152) |
| KC3 | 80.5% | 78% (82.5%) | 78% (80.5%) | 78% (82.5%) | 78% (82.5%) | 82% (84%) | 0.195 (0.16) |
| MC1 | 99.36% | 99.01% (99.4%) | 99.01% (99.4%) | 99.01% (99.4%) | 99.01% (99.44%) | 99.32% (99.39%) | 0.006 (0.006) |
| MC2 | 62.99% | 57.48% (66.14%) | 57.48% (66.14%) | 57.48% (62.99%) | 57.48% (62.99%) | 64.56% (66.14%) | 0.370 (0.338) |
| MW1 | 89.39% | 86.36% (90.53%) | 86.36% (90.53%) | 86.36% (92.05%) | 86.36% (92.05%) | 88.63% (90.53%) | 0.106 (0.094) |
| PC1 | 90.9% | 88.93% (92.09%) | 88.93% (92.09%) | 88.93% (91.3%) | 88.93% (91.3%) | 91.17% (91.43%) | 0.09 (0.085) |
| PC2 | 98.73% | 97.98% (98.86%) | 97.98% (98.67%) | 97.98% (98.99%) | 97.98% (98.99%) | 98.67% (98.99%) | 0.012 (0.010) |
| PC3 | 85.77% | 84.17% (86.13%) | 84.17% (85.86%) | 84.17% (87.82%) | 84.17% (87.82%) | 86.66% (87.55%) | 0.142 (0.124) |
| PC4 | 89.56% | 89.7% (89.84%) | 89.7% (88.27%) | 89.7% (89.84%) | 89.7% (89.84%) | 88.56% (88.92%) | 0.104 (0.110) |
| PC5 | 97.35% | 97.24% (97.38%) | 97.24% (97.29%) | 97.24% (97.38%) | 97.24% (97.38%) | 97.15% (97.34%) | 0.026 (0.026) |

### *9.2 Conclusion*

The conclusions that can be drawn from the results of Table 22 and Table 23 using a probabilistic and a non probabilistic classifiers on the MDP datasets confirm the main trends characterizing the evaluations of Tables 4, 5, 6.

## References


[1] David W. Aha, D. Kibler, and Marc K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6(1):37–66, January 1991.

[2] N. Ben Amor, S. Benferhat, and Z. Elouedi. Qualitative classification and evaluation in possibilistic decision trees. In *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2004, Budapest, Hungary, July 25-29, 2004.*, pages 653–657, 2004.

[3] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99–15, Chalmers Univ., March 2000.

[4] N. Ben Amor, S. Benferhat, and Z. Elouedi. Naive Bayes vs Decision Trees in Intrusion Detection Systems. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 420–424, New York, NY, USA, 2004. ACM.

[5] N. Ben Amor, S. Benferhat, and Z. Elouedi. Towards a definition of evaluation criteria for probabilistic classifiers. In *Eighth European Conference of Symbolic and Quantitative Approaches to Reasoning with Uncertainty ECSQARU-2005*, pages 921–931. LNAI 3571, Springer Verlag, 2005.

[6] S. Benferhat, A. Boudjelida, K. Tabia, and H. Drias. An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge. *Appl. Intell.*, 38(4):520–540, 2013.

[7] S. Benferhat, T. Kenaza, and A. Mokhtari. Tree-augmented naive bayes for alert correlation. In *3rd conference on Advances in Computer Security and Forensics(ACSF'08)*, pages 45–52, jul 2008.

[8] S. Benferhat and K. Sedki. Alert correlation based on a logical handling of administrator preferences and knowledge. In *International Conference on Security and Cryptography(SECRYPT'08)*, pages 50–56, Porto, Portugal, jul 2008.

[9] S. Benferhat and K. Sedki. An alert correlation approach based on security operator's knowledge and preferences. *Journal of Applied Non-Classical Logics*, 20(1-2):7–37, 2010.
[10] S. Benferhat and K. Tabia. Classification features for detecting server-side and client-side web attacks. In *IFIP TC-11 23rd International Information Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008, September 7-10, 2008, Milano, Italy*, pages 729–733, 2008.
[11] Z. Bin and A. Ghorbani. Alert correlation for extracting attack strategies. *I. J. Network Security*, 3(3):244–258, 2006.
[12] Nitesh V. Chawla. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*, pages 875–886. 2010.
[13] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, Jan 1970.
[14] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 09(4):309–347, October 1992.
[15] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1 edition, 2000.
[16] F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy*, pages 187–200, USA, 2002.
[17] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer.
[18] H. T. Elshoush and I. M. Osman. Alert correlation in collaborative intelligent intrusion detection systemsa survey. *Applied Soft Computing*, 11(7):4349 – 4365, 2011.
[19] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.
[20] M. Gebel and C. Weihs. Calibrating classifier scores into probabilities. In Reinhold Decker and Hans-J. Lenz, editors, *Advances in Data Analysis*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 141–148. Springer Berlin Heidelberg, 2007.
[21] Kenneth L. Ingham and Hajime Inoue. Comparing anomaly detection techniques for http. In *Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection*, RAID'07, pages 42–62, Berlin, Heidelberg, 2007. Springer-Verlag.
[22] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 366–375, New York, NY, USA, 2002. ACM.
[23] L. Kufel. Security event monitoring in a distributed systems environment. *Security Privacy, IEEE*, 11(1):36–43, Jan 2013.
[24] I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
[25] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
[26] B. Lerner and R. Malka. Investigation of the k2 algorithm in learning bayesian network classifiers. *Applied Artificial Intelligence*, 25(1):74–96, 2011. n/a.
[27] Z. Mian. *Network Intrusion Detection: Monitoring, Simulation and Visualization*. PhD thesis, Orlando, FL, USA, 2005. AAI3188144.
[28] S. A. Mirheidari, S. Arshad, and R. Jalili. Alert correlation algorithms: A survey and taxonomy. In *CSS*, volume 8300 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2013.
[29] P. Ning, Y. Cui, and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *9th ACM conference on Computer and communications security*, pages 245–254, NY, USA, 2002. ACM.
[30] A. Patcha and J. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, 2007.
[31] Z. Qin, C. Zhang, T. Wang, and S. Zhang. Cost sensitive classification in data mining. In *Advanced Data Mining and Applications*, volume 6440 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg, 2010.
[32] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.
[33] M. Roesch. Snort - lightweight intrusion detection for networks. pages 229–238, 1999.
[34] L. Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046 – 4072, 2009.
[35] R. Smith, N. Japkowicz, M. Dondo, and P. Mason. Using unsupervised learning for network alert correlation. In *21st conference on Advances in artificial intelligence*, pages 308–319, Berlin, Heidelberg, 2008. Springer-Verlag.
[36] G. C. Tjhai, M. Papadaki, S. Furnell, and N. L. Clarke. Investigating the problem of ids false alarms: An experimental study using snort. In *23rd International Information Security Conference SEC 2008*, pages 253–267, 2008.
[37] E. Tombini, H. Debar, L. Mé, and M. Ducassé. A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic. In *Annual Computer Security Applications Conference 2004*, pages 428–437, Beijing Chine, 12 2004.
[38] P. K. Turaga, R. Chellappa, and A. Veeraraghavan. Advances in video-based human activity analysis: Challenges and approaches. *Advances in Computers*, 80:237–290, 2010.
[39] Marcel v. G. and J. F. Lucas. Using background knowledge to construct bayesian classifiers for data-poor domains. In *Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Queens' College, Cambridge, UK, 13-15 December 2004*, pages 269–282. Springer.
[40] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Recent Advances in Intrusion Detection*, pages 54–68, London, UK, 2001. Springer-Verlag.

**Mouaad Kezih** received his M. Sc. Degrees in Multimedia and Digital Communication from Annaba University, Algeria, in 2010. He is currently pursuing a PhD. degree in Multimedia and Digital Communication at Badji Mokhtar Annaba University, Algeria. His research interests include Artificial Intelligence, Computer security, intrusion detection and alert correlation.

**Mahmoud Taibi** received his BSc in electrical engineering from USTO University, Oran, Algeria in 1980, then an MSc from Badji-Mokhtar University, Annaba, Algeria in 1996. Currently, he is a full professor in computer science since 2006 at Badji-Mokhtar University of Annaba, Algeria. His research interests are in intelligent systems, security and for medical applications as well as document analysis.

**Salem Benferhat** is a full professor at University of Artois and CRIL laboratory (CNRS UMR 8188). His current research interests are in knowledge representation, uncertainty handling, possibilistic networks, causality, non-classical logics, intrusion detection and alerts correlation. He is the author or the co-author of a large number of research papers (more than 250 papers including 45 international journal papers and 155 papers in international conferences).

**Karim Tabia** is assistant professor at the University of Artois since September 2010. He finished his PhD thesis in 2008 at CRIL - Artois University on Graphical models and anomaly-based approaches for intrusions detection. His research topics revolve around artificial intelligence, knowledge representation and reasoning under uncertain and imperfect information. The application domains of his works are mainly in computer security. He co-authored 6 publications in well-rated international journals. He is also the author or co-author of 21 papers in international conference including AI and computer security.