Extensions, Analysis and Experimental Assessment of a Probabilistic Ensemble-learning Framework for Detecting Deviances in Business Process Instances

Alfredo Cuzzocrea[1,2], Francesco Folino[1], Massimo Guarascio[1] and Luigi Pontieri[1]
[1]*ICAR-CNR, Via Pietro Bucci 7/11C, I-87036, Rende (CS), Italy*
[2]*DIA Department, University of Trieste, I-34127, Trieste, Italy*

Keywords: Business Process Intelligence, Classification, Deviance Detection.

Abstract: The problem of discovering an effective Deviance Detection Model (DDM) out of log data, has been attracting increasing attention in recent years in the very active research areas of *Business Process Intelligence* (BPI) and of *Process Mining*. Such a model can be used to assess whether novel instances of the business process are deviant or not, which is a hot topic in many application scenarios such as cybersecurity and fraud detection. This paper extends a previous proposal where an innovative ensemble-learning framework for mining business process deviances was introduced, hinging on multi-view learning scheme. Specifically, we introduce here an alternative meta-learning method for probabilistically combining the predictions of different base DDMs. The entire learning method is embedded into a conceptual system architecture that is meant to support the detection and analysis of deviances in a Business Process Management scenario. We also discuss a wide and comprehensive experimental analysis of the proposed approach and of a state-of-the-art DDM discovery solution. The experimental findings confirm the flexibility, reliability and effectiveness of the proposed deviance detection approach, and the improvement gained over its previous version.

1 INTRODUCTION

Business Process Intelligence (BPI) and Process Mining are active areas of research, which enjoy many relevant real-life applications.

A recent thread of research in this area concerns the problem of detecting deviant instances of a business process (a.k.a. "deviance mining"), i.e. instances that *deviate* from normal outcomes (Suriadi et al., 2013; Bose and van der Aalst, 2013; Nguyen et al., 2014; Lo et al., 2009; Cuzzocrea et al., 2015). This problem is relevant in many application scenarios such as cybersecurity and fraud detection, and so forth. Essentially, the problem has been rephrased in the literature as a binary classification problem, where the class of all deviant process instances is to be discriminated from the one gathering all the other (normal) instances of the process under analysis. This problem has been faced by inducing a suitable classification model (named hereinafter *Deviance Detection Model*, or *DDM* for short), out of some flat representation of a historical log of process traces (labelled each as either deviant or normal). Such a model can be then applied to any new instance of the process to estimate whether it is a deviance or not.

In our opinion, the current literature in the field has not fully addressed a series of issues that are likely to arise in many real application scenarios. We discuss them in the following.

I1 First of all, most of the deviance mining approaches (including, in particular, (Bose and van der Aalst, 2013; Swinnen et al., 2011; Nguyen et al., 2014)) rely on training a single DDM from a propositional view of the given log, where each trace is encoded into a fixed-length form by projecting the associated sequence of log events onto some given set of behavioral patterns (such as the *individual activities, maximal repeats, or tandem repeats* used in (Bose and van der Aalst, 2013; Nguyen et al., 2014) and the experiments discussed in Section 5). As shown in (Nguyen et al., 2014) using multiple heterogenous kinds of pattern can lead to higher classification accuracy. However, mixing all heterogeneous patterns into a single view is likely to produce a high-dimensional, sparse and redundant representation

Cuzzocrea, A., Folino, F., Guarascio, M. and Pontieri, L.
Extensions, Analysis and Experimental Assessment of a Probabilistic Ensemble-learning Framework for Detecting Deviances in Business Process Instances.
DOI: 10.5220/0006340001620173
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 162-173
ISBN: 978-989-758-247-9

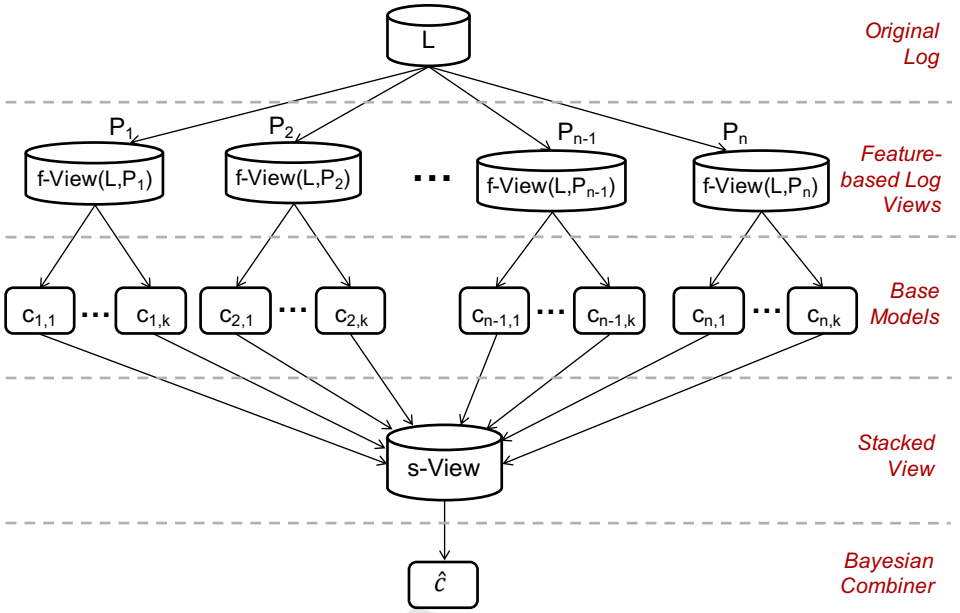

Figure 1: Conceptual data-processing flow of the proposed approach: original data, transformed data and discovered DDMs.

of the training instances. This calls for adopting some multi-view learning strategy (like the one proposed in (Cuzzocrea et al., 2015)), capable of exploiting different complementary views of the given traces.

I2 In many real-life applications, analyzing a presumably deviant case is an expensive task. If equipping the detected deviant instances with a deviance probability score, this task could be focused on more suspicious instances. This capability could allow to use deviance prediction tools more flexibly and more effectively in real BPM systems.

I3 Deviance mining analyses must be often carried out in situations where the deviant instances are far less than normal ones. This is a case of "class imbalance" (Japkowicz and Stephen, 1997), which constitutes a challenge for most classifier induction approaches, conceived to maximize the overall accuracy without paying special attention to the minority (i.e. deviant, in our case) instances.

Some of these issues have been faced in a previous work of ours (Cuzzocrea et al., 2015), where an ensemble-learning approach was proposed that exploits multi-view learning to solve the deviance mining problem. This approach is summarized pictorially in Figure 1. Basically, a number, say n, of complementary feature-based views of the given log L are produced, which provide each a vector-space encoding of both context properties and behavioral patterns of the traces in L. Two layers of deviation-detection models are induced from these views: *(i)* a collection of base models (learnt by applying k different classifier-induction methods to one of the n views above), and *(ii)* a meta model, which integrates the predictions of the base models into a "high-order" deviance forecast. As explained in (Cuzzocrea et al., 2015), the latter model deals with heterogeneous behavioral patterns at a higher level of abstraction, so effectively addressing the issue *I1*.

In order to deal with situations where deviant instances are far less than normal ones (issue *I3*), the learning procedure can be made integrate a resampling mechanism, in order to reduce the level of class imbalance.

In this paper, we provide three novel and significant contributions over (Cuzzocrea et al., 2015), namely:

- two alternative Bayesian meta-learning methods for probabilistically combining the predictions of different base DDMs;
- a conceptual system architecture supporting the detection and analysis of deviances in a Business Process Management scenario; and
- a wide and comprehensive experimental analysis of algorithm `HO-DDM-mine` and of a state-of-the-art method (Nguyen et al., 2014).

More specifically, in our current approach the meta-learning task can take advantage of (a customized version of) one of the following methods for the induction of probabilistic classifiers:

AODE (Webb et al., 2005) or *HNB* (Zhang et al., 2005). Notably, both these methods relax the assumption of attribute independence that underlies Naïve Bayes models —this assumption hardly holds, indeed, in our DDM learning setting, where several activities/patterns are likely to be correlated one another— with negligible additional costs in terms of computation time. This allows to associate any new process instance with a reliable estimate of the probability that it is deviant, thus addressing the delicate issue *I2*.

The results of our empirical analysis confirmed the flexibility, reliability and effectiveness of the proposed deviance detection approach, and the improvement gained over its previous version.

The rest of the paper is organized as follows. Section 2 introduces some basic concepts and notation on the specific kinds of Bayesian classifiers that are used in our approach to accomplish the meta-learning subtask. The specific kind of overall DDM (named *High-Order Deviance Detection Model*, short `HO-DDM`) that is eventually returned by our approach is illustrated in Section 3. Section 4 then presents a conceptual system architecture for the detection and analysis of deviant process instances, in a BPM scenario, summarizes the induction algorithm for extracting a `HO-DDM` out a given set of historical log traces. After discussing, in Section 5, the results of the experimental activities that we have conducted on a real case study, we draw a few concluding remarks in Section 6.

2 PRELIMINARIES: BAYESIAN MODELS

As explained before, we want to probabilistically classify a process trace as either deviant or not, in order to equip the trace with a measure of confidence in the fact that it is really a deviance. In order to do this in a scalable way, we resort to two extensions of the popular Naïve Bayes method. Before presenting these methods in details, let us introduce some basic concepts and notation concerning Bayesian classifiers, in general, and Naïve Bayes models, in particular.

In general, *Bayesian classifiers* combine a priori knowledge of the classes with new evidence gathered from data. Let us consider an instance space with $m+1$ nominal attributes $X_1,\ldots,X_m,Y$, such that $m \in \mathbb{N}$, and a class attribute Y encoding the class label (if known) of any instance. For any attribute $Z \in \{X_1,\ldots,X_m,Y\}$, let $dom(Z)$ be the associated domain, and let $dom(Y) = \{c_1,\ldots,c_k\}$. Given an instance $\mathbf{x} = \langle x_1,\ldots,x_m\rangle$, where x_i is the value observed for attribute X_i, the classification problem amounts to estimating the class label $y \in dom(Y)$ for $\mathbf{x}$, based on some suitable classification model, previously extracted from (already classified) training instances.

In a hard classification setting, a Bayesian classifier computes $P(y|\mathbf{x})$ for each class $y \in dom(Y)$, and assigns $\mathbf{x}$ to the class associated with the with the highest probability, i.e., $y^* = \arg\max_{y\in dom(Y)} (P(y|\mathbf{x}))$. Clearly, $P(y|\mathbf{x}) = P(y,\mathbf{x})/P(\mathbf{x})$ and $\arg\max_{y\in dom(Y)} (P(y|\mathbf{x})) = \arg\max_{y\in dom(Y)} P(y,\mathbf{x})$. As $P(y,\mathbf{x}) = P(y)\cdot P(\mathbf{x}|y)$, this hard classification task amounts to finding $y^* \in dom(Y)$ such that: $y^* = \arg\max_{y\in dom(Y)} (P(y)\cdot P(\mathbf{x}|y))$.

For a fully probabilistic classification, every class membership probability can be estimated as:

$$P(y|\mathbf{x}) \approx \frac{P(y,\mathbf{x})}{\sum_{y'\in dom(Y)} P(y',\mathbf{x})} = \frac{P(y,\mathbf{x})}{\sum_{y'\in dom(Y)} P(y')\cdot P(\mathbf{x}|y')} \quad (1)$$

While the prior probability $P(y)$ can be derived from the sample frequencies in the training set, one should estimate the conditional probability $P(\mathbf{x}|y)$. In Naïve Bayes (NB) classifiers the latter task is accomplished by assuming that all the attributes are conditionally independent of one another, given the class label y. Under this hypothesis, for each $y \in dom(Y)$, it is $P(y,\mathbf{x}) = P(y)\cdot\prod_{i=1}^{m} P(x_i|y)$. Thus, the instance $\mathbf{x}$ is assigned to the class $y^* \in dom(Y)$ such that $y^* = \arg\max_{y\in dom(Y)} (P(y)\cdot\prod_{i=1}^{m} P(x_i|y))$. Any "soft" class membership probability $P(y|\mathbf{x})$ can be computed in a similar way.

NB classifiers work well in a wide range of applications (Langley et al., 1992). However, in certain settings (like the one considered in this work), the attribute independence assumption is inappropriate. Many efforts have been made in the literature to relax such an assumption, without making the probability distribution too complex and hard to induce. We next briefly illustrate two popular extensions of NB classifiers, namely AODE and HNB, which are both used in the current implementation of our approach.

2.1 AODE

One-dependence estimators (ODEs) (Sahami, 1996) generalize NBs by allowing each attribute to depend on one other attribute besides the class. A subclass of ODEs are SPODEs (Keogh and Pazzani, 2002; Ying et al., 2007), where all the attributes can only depend on a single common one, the *super-parent* (in addition to the class). For example, for a SPODE with super-parent X_p, $P(y,\mathbf{x})$ is computed as follows:

$$P(y,\mathbf{x}) = P(y,x_p)\cdot P(\mathbf{x}|y,x_p) = P(y,x_p)\cdot\prod_{i=1}^{m} P(x_i|y,x_p)$$

SPODEs are typically combined into a sort of ensemble, in order to reduce the classification variance and ensure higher accuracy (Webb et al., 2005). Clearly, if the data instances feature m attributes, at most m different SPODEs can be combined. The different SPODE models in such an ensemble are usually merged by computing an overall probability estimate for $P(y,\mathbf{x})$ as a linear combination of the probability estimates returned by all of these "base" models:

$$P(y,\mathbf{x}) = \sum_{j=1}^{m} w_j \cdot P_j(y,\mathbf{x}) = \sum_{j=1}^{m} w_j \cdot P(y,x_j)\cdot\prod_{i=1}^{m} P(x_i|y,x_j) \quad (2)$$

where $P_j(y,\mathbf{x})$ is the estimate of j-th SPODE for $\mathbf{x}$, and w_j is the weight given to the same SPODE.

The *Averaged One-Dependence Estimators* (AODE) (Webb et al., 2005) method adopts a simple combination strategy, which only considers the super-parents that appear in the training dataset more than a minimum support threshold $h \in \mathbb{N}$, and assigns uniform weights to all of their corresponding SPODEs. Hereinafter, we simply keep fixex $h = 1$ —notice that the same choice has been made in the experiments described in Section 5. According to this strategy, $P(y,\mathbf{x})$ is eventually estimated as follows:

$$P(y,\mathbf{x}) = \frac{\sum_{j\in\mathcal{S}} P(y,x_j)\cdot\prod_{i=1}^{m} P(x_i|y,x_j)}{|\mathcal{S}|} \quad (3)$$

where $\mathcal{S}$ is the subset of values x_j that occur at least h times in the training set (and w_j is fixed to $1/|\mathcal{S}|$ for all x_j in $\mathcal{S}$).

In order to estimate, for each tuple $\mathbf{x} \in \mathbf{X}$ and each possible class label $y \in dom(Y)$, the probability $P(y|\mathbf{x})$, one can simply normalize the numerator in Eq. 3 over all the classes as follows:

$$P(y|\mathbf{x}) = \frac{P(y,\mathbf{x})}{\sum_{y'\in Y} P(y',\mathbf{x})} = \frac{\sum_{j\in\mathcal{S}} P(y,x_j)\cdot\prod_{i=1}^{m} P(x_i|y,x_j)}{\sum_{y'\in dom(Y)}\sum_{j\in\mathcal{S}} P(y',x_j)\cdot\prod_{i=1}^{m} P(x_i|y',x_j)} \quad (4)$$

2.2 Hidden Naïve Bayes

The basic assumption underlying a SPODE is that each attribute can only depend on another one (i.e. the super-parent), even though several other attributes might actually influence the former. This assumption is mitigated by averaging different SPODEs. By contrast, in the *Hidden Naïve Bayes* (*HNB* for short) model, each attribute x_i is assumed to be possibly influenced by a artificial "hidden" parent attribute, capturing the influence that all other attributes may have on x_i. In other words, each attribute x_i is allowed to depend on a hidden parent x_i^h ($\forall i = 1,\ldots,m$), in addition to the class y. Therefore, the probability $P(y,\mathbf{x})$ computed by an HNB can be written as:

$$P(y,\mathbf{x}) = P(y)\cdot\prod_{i=1}^{m} P(x_i|y,x_i^h) = P(y)\cdot\prod_{i=1}^{m}\left(\sum_{j=1,j\neq i}^{m} w_{ij}\cdot P(x_i|y,x_j)\right) \quad (5)$$

where $\sum_{j=1}^{m} w_{i,j} = 1$.

Clearly, the hidden parent x_i^h for x_i is defined as a mixture of the weighted influences from all other attributes.

A key point in the construction of an HNB is the definition of the weights $w_{i,j}$ (for any i,j in $\{1,\ldots,m\}$). In the approach proposed in (Zhang et al., 2005), any weight $w_{i,j}$ is computed on the basis of the *Conditional Mutual Information* $I_P(x_i,x_j|y)$ between x_i and x_j, which is defined as follows:

$$I_p(x_i,x_j|y) = \sum_{x_i,x_j,y} P(x_i,x_j,y)\cdot\log\frac{P(x_i,x_j|y)}{P(x_i|y)P(x_j|y)} \quad (6)$$

More specifically, for any $i,j \in \{1,\ldots,m\}$, it is:

$$w_{i,j} = \frac{I_p(x_i,x_j|y)}{\sum_{j=1,j\neq i} I_P(x_i,x_j|y)} \quad (7)$$

The posterior class-membership probability $P(y|\mathbf{x})$, for each tuple $\mathbf{x} \in \mathbf{X}$ and each possible class $y \in dom(Y)$, can be simply obtained as follows:

$$P(y|\mathbf{x}) = \frac{P(y,\mathbf{x})}{P(\mathbf{x})} = \frac{P(y,\mathbf{x})}{\sum_{y'\in dom(Y)} P(y',\mathbf{x})} = \frac{P(y)\cdot\prod_{i=1}^{m}\left(\sum_{j=1,j\neq i}^{m}(w_{ij}\cdot P(x_i|y,x_j)\right)}{\sum_{y'\in dom(Y)}\left(P(y')\cdot\prod_{i=1}^{m}\left(\sum_{j=1,j\neq i}^{m}(w_{ij}\cdot P(x_i|y',x_j)\right)\right)} \quad (8)$$

3 HIGH-ORDER DEVIANCE DETECTION MODELS

Our approach to the discovery of an HO-DDM relies on training multiple base learners on different feature-based views of a given log L. According to the processing flow depicted in Figure 1, the approach founds on training multiple base learners on n different feature-based views of L, produced each according to a different list of behavioral patterns (like those used in (Bose and van der Aalst, 2013; Nguyen et al., 2014; Cuzzocrea et al., 2015), and in our experimentation).

Let $P_1,\ldots,P_n$ these pattern lists and $f\text{-}View(L,P_1),\ldots,f\text{-}View(L,P_n)$ be the feature-based views of L that are obtained by encoding the traces in L according to the patterns in $P_1,\ldots,P_n$, respectively. More precisely, for each P_i and each $\tau \in L$, the view $f\text{-}View(L,P_i)$ contains a tuple $f\text{-}View(\tau,P_i)$ that encodes all the context data of τ and the projection of τ onto the patterns of P_i. In particular, the correlation between τ and each pattern $p_j \in P_i$ is computed as the number of times that p_j occurs in τ. Further details can be found in (Cuzzocrea et al., 2015)

By applying k different learning algorithms to all of these views, a list CL of $r = n \times k$ "base" DDMs is obtained. These multi-view base classifiers are made undergo a stacking-oriented meta-learning scheme, in order to integrate all of them into a single higher-level probabilistic DDM.

For the sake of notation, let PL be a list, of the same length as CL, of pattern lists such that $PL[q]$ is the specific list of patterns that was used to train the model $CL[q]$, for any $q \in \{1,\ldots,r\}$.[1]

Then the meta-learning task is performed on a meta-view $s\text{-}View(L,CL,PL)$ of log L that stores, for each trace $\tau \in L$, a tuple $s\text{-}View(\tau,CL,PL)$ having the same class label as τ, and featuring (as input attributes) both the predictions of all the DDMs in CL and the case/context attributes of τ. From $s\text{-}View(L,CL,PL)$ a probabilistic meta-classifier (one among *HNB* (Zhang et al., 2005) and *AODE* (Webb et al., 2005)) is eventually induced, which can combine the predictions of all of the base DDMs in CL.

The final result of this learning procedure is a multi-view deviance detection model, named *High-Order Deviation Detection Model* (HO-DDM), which is formally defined below.

Definition 1 (HO-DDM) Let L be a log over some proper trace universe $\mathcal{T}$, and $PROP(\mathcal{T})$ be the space of all the data attributes that are associated with the traces of $\mathcal{T}$. Then, a *High-Order Deviance Detection Model* (HO-DDM) for L is a triple of the form $H = \langle CL, PL, \hat{c} \rangle$, where: *(i)* PL is a list of r pattern lists, for some $r \in \mathbb{N}$; *(ii)* CL is a list of r (base) DDMs such that, for each $i \in \{1,\ldots,r\}$, $CL[i]$ is a DDM, learnt by using $f\text{-}View(L,PL[i])$ as training set, encoding a function of the form $CL[i] : PROP(\mathcal{T}) \times \mathbb{R}^{|P_i|} \rightarrow \{0,1\}$, which maps the propositional representation $f\text{-}View(\tau,PL[i])$ of any trace $\tau \in \mathcal{T}$ to a class label in $\{0,1\}$; and *(iii)* $\hat{c} : PROP(\mathcal{T}) \times \{0,1\}^r \rightarrow [0,1]$ is a (meta) classifier providing an estimate for the probability that any trace τ in $\mathcal{T}$ is deviant, based on its ("stacking-oriented") representation $s\text{-}View(\tau,CL,PL)$. □

In the current implementation of our approach, the predictions of all the discovered base DDMs in CL are combined using a probabilistic classifier $\hat{c}$, computed by applying one of the two Bayesian learning methods described in the previous section. When a novel trace τ is to be classified, each base model $CL[i]$ in the ensemble is applied to the vector-space representation $f\text{-}View(\tau,PL[i])$ of τ (produced according to the same list $PL[i]$ of patterns that was used to induce $CL[i]$). The predictions made of the models in CL are then combined into a single prediction by $\hat{c}$. Specifically, by providing the latter model with a propositional view of τ mixing the original data properties of τ (stored in $prop(\tau)$) and the predictions assigned to τ by the base models, the deviance probability score $\hat{c}(\tau)$ is eventually returned as output. A simple way to classify τ as deviant is to check whether $\hat{c}(\tau) > 0.5$ —in this case to τ is assigned the label "1". For the sake of flexibility, we allow for fixing a lower deviance probability threshold $\gamma \in (0,1)$, so that any τ is deemed as deviant if and only if $\hat{c}(\tau) > \gamma$.

4 DEVIANCE MINING AND ANALYSIS FRAMEWORK: ARCHITECTURE AND IMPLEMENTATION

Figure 2 illustrates a conceptual system architecture supporting the detection and analysis of deviant process instances. Such a system, following a multi-layer structure, is meant to handle the whole flow of log-based process analysis, from the discovery of an HO-DDM out of historical (labeled) log data, to the detection of new deviant process instance, and the inspection and analysis of both the discovered models and detected deviances. The system is composed of four different layers: *Data/Model Repository*, *De-*

[1] With regard to the the labelling scheme of Figure 1 (and assuming that the base DDMs appear in CL in the same left-to-right order as in the figure), it is $CL[q] = c_{i,j}$ iff $q = (i-1) \times k + j$ and $PL[q] = P_i$ iff $i = \lfloor (q-1)/k \rfloor + 1$.

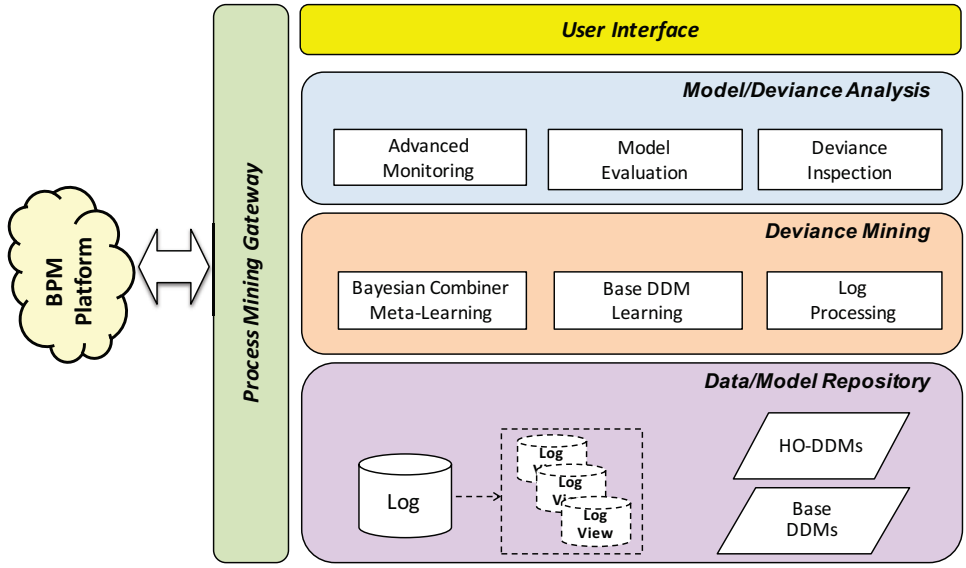

Figure 2: System Architecture for HO-DDM-mine.

viance Mining, *Model/Deviance Analysis*, and *User Interface*.

Essentially, the bottommost layer is responsible for storing both historical process logs and different kinds of views (namely f-Views and s-Views) extracted from them, as well as the different kinds of models that compose our HO-DDMs. All the process mining and preprocessing methods implemented in the Deviance Mining layer are meant to support the induction of an HO-DDM out of historical log traces, annotated each with a boolean deviance flag. The discovery of a HO-DDM from a given log L can be accomplished with the algorithm HO-DDM-mine proposed in (Cuzzocrea et al., 2015), which is summarized in the following. Basically, the algorithm follows a two-phase computation strategy. In the first phase, a number of base classifiers are discovered by applying a given set of inductive learning methods to different views of L, obtained each by projecting the traces in L onto a different space of features. In the second phase, all of these base classifiers are combined into a single DDM, based on a meta-learning (stacking) procedure.

In more detail, the algorithm iterates over the following main steps:

1. Preliminary to the extraction of the patterns, in case log L is imbalanced, it can be made undergo an oversampling procedure, where each deviant trace in L is simply duplicated a certain number of times.
2. For each pattern family specified by the analyst (e.g., *individual activities*, *tandem repeats*, *maximal repeats* (Bose and van der Aalst, 2013; Nguyen et al., 2014; Cuzzocrea et al., 2015)), a list of relevant patterns of that family are extracted from L. The patterns that can be generated for each family are selected according to a frequency-based strategy, where the patterns with a high enough frequency in the log (or the q top frequent ones) are kept.
3. For each pattern list P of those that have been extracted in the previous step, the feature-based view $f\text{-}View(L,P)$ is materialized, and used as training set for discovering different base DDMs (using a different base learning algorithm for inducing each of the latter). As a result, a list CL of base DDMs is obtained, and an associated list PL of pattern lists (such that $PL[i]$ is the pattern list used to train $CL[i]$, for any position i in CL).
4. The discovered DDMs in CL are then combined into a single overall meta-classifier using a stacking strategy. To this end, the "stacked" view $s\text{-}View(L,CL,PL)$ is computed, and a meta-classifier $\hat{c}$ is induced from it, by applying one of the two alternative probabilistic learning methods (e.g., our customized versions of the methods AODE or HNB described in Section 2). The discovered combiner must be able to compute, for any "stacked" tuple $\mathbf{x} \in X$, an estimate $\hat{c}(\mathbf{x})$ of $\mathbf{x}$'s deviance probability (i.e. $\hat{c}(\mathbf{x}) \approx P(Y = 1|\mathbf{x})$) according to Eq. 4 or 8 This can be done efficiently by pre-computing a number of sufficient statistics, as described later on in this section.

All the discovered DDMs (be them base models or HO-DDMs) are made available to the

Model/Deviance Analysis Layer, which can produce a series of evaluation measures, like those used in our experimentations to quantitatively assess the validity of our approach. The *Advanced Monitoring* module offers deviance-oriented runtime-support services, which mainly consist in the notification of alert messages and in the suggestion of possible countermeasures to be possibly undertaken when a deviant process instance is detected. Finally, the *Deviance Inspection* module is meant to allow the analyst to inspect and study single process instances detected as (mostly) deviant, and try to understand the causes of their abnormal behavior.

Details on the discovery of a `HO-DDM`: statistics underlying the Bayesian combiners. As explained previously, the Bayesian combiner $\hat{c}$ in any `HO-DDM` (cf. Def. 1) must return, for each "stacked" tuple $\mathbf{x} \in X$, an estimate $\hat{c}(\mathbf{x})$ of $\mathbf{x}$'s deviance probability (i.e. $\hat{c}(\mathbf{x}) \approx P(Y = 1|\mathbf{x})$) according to Eq. 4 or 8. Such an estimate is computed on the basis of the following (precomputed) statistics, which are all derived from *s-View*(L,CL,PL) for all $i, j \in \{1,\ldots,m\}$, $x_i \in dom(X_i)$, and $x_j \in dom(X_j)$, and constitute the backbone of the combiner model $\hat{c}$:

- n_i is the number of values for attribute X_i;
- D (resp. N) is the number of deviant (resp. normal) tuples in *s-View*(L,CL,PL);
- D_j (resp. N_j) is the number of deviant (resp. normal) tuples in *s-View*(L,CL,PL) that feature the value x_j,
- D_{ji} (resp. N_{ji}) is the number of deviant (resp. normal) tuples in *s-View*(L,CL,PL) that feature both values x_j and x_i,
- K is the number of tuples in *s-View*(L,CL,PL) (all having a known value of the deviance label),
- K_j is the number of tuples in *s-View*(L,CL,PL) for which the value of attribute X_j is known,
- K_{ji} is the number of tuples in *s-View*(L,CL,PL) for which the values of both attributes X_i and X_j are known. □

Notice that the counters above are sufficient to estimate every base probability of the form $P(y)$, $P(y,x_j)$ and $P(y,x_j,x_i)$. The latter, in turn, can be exploited to estimate every probability of the form $P(x_i|y)$ and $P(x_i|y,x_j)$ —for each values x_j and x_i of the attributes X_j and X_i (with the former playing as super-parent), for $i, j \in \{1,\ldots,m\}$ and $y \in \{0,1\}$— that are needed to compute $P(y|\mathbf{x})$ according to Eq. 4 and 8. To make the estimation more robust, these base probabilities are computed by using the *Laplace estimation* method (Webb et al., 2005; Zhang et al., 2005), as specified in the following: $P(Y = 1) = \frac{D+1}{K+2}$; $P(Y = 1,x_j) = \frac{D_j+1}{K_j+2\cdot n_j}$; $P(Y = 1,x_j,x_i) = \frac{D_{ji}+1}{K_{ji}+2\cdot n_i\cdot n_j}$; $P(Y = 0) = \frac{N+1}{K+2}$; $P(Y = 0,x_j) = \frac{N_j+1}{K_j+2\cdot n_j}$; $P(Y = 0,x_j,x_i) = \frac{N_{ji}+1}{K_{ji}+2\cdot n_i\cdot n_j}$; $P(x_i|Y = 1) = P(Y = 1,x_i)/P(Y = 1)$; $P(x_i|Y = 0) = P(Y = 0,x_i)/P(Y = 0)$; $P(x_i|Y = 1,x_j) = P(Y = 1,x_j,x_i)/P(Y = 1,x_j)$; $P(x_i|Y = 0,x_j) = P(Y = 0,x_j,x_i)/P(Y = 0,x_j)$.

5 EXPERIMENTAL ANALYSIS

The capability of our approach to effectively recognize deviant behaviors has been assessed by conducting a series of tests on a real-life log, storing information on the clinical pathways of gynecologic cancer patients within a Dutch hospital. This log was made available as a benchmark dataset for the *2011 BPI Challenge* (van Dongen, 2011). Details on these datasets, omitted here for the sake of space, can be found in (Cuzzocrea et al., 2015).

Different evaluation metrics exist in the literature for testing the effectiveness of classification models in the presence of a rare class. Carefully choosing these metrics is important since the usage of metrics that do not adequately account for the rarity of the minority class may easily lead to overestimating the accuracy of a classifier. Specifically, we used four metrics widely used over imbalanced data, i.e. *area under the ROC curve* (*AUC*) (Bradley, 1997), the *G-mean* (Kubat et al., 1997), *Precision* (*P*), and *Recall* (*R*) (Buckland and Gey, 1994).

5.1 Settings

Testing our approach requires two key settings: (1) the kind of patterns used to project the log traces onto a vector space, and (2) the classifier-induction methods employed to derive, from such a feature-based representation of the traces, the base and combined models that compose the overall `HO-DDM`.

As concerns the former point, as a first family of behavioral patterns, denoted by `IA` (i.e. *individual activities*), we simply considered all the process activities in their own. In this case, for any trace, we regard each activity, say a, as an additional (pattern-oriented) feature of the trace, storing the number of times that a occurs in the trace. In order to produce more sophisticated representations of traces' behaviors, we also considered (as done in (Bose and van der Aalst, 2013; Nguyen et al., 2014)) all the sequence-based patterns possibly capturing control-flow constructs (e.g., subprocesses, loops, and parallelism) ruling the behav-

Table 1: Prediction results obtained on the $BPIC11_{CC}$ log when using different configurations of the (single DDM) competitor method (Nguyen et al., 2014). All the values were computed by averaging the results of 5 trials, performed according to a 5 fold cross-validation scheme. For each metrics, the best outcome is reported in bold.

Alg.	*Patterns*	*AUC*	*G-Mean*	*R*	*P*	*AvgRank*	*AvgRank_5%*
IBk	{IA.AMR}	0.771±0.019	0.538±0.049	0.321±0.062	0.458±0.106	9.00	2.75
	{IA.ATR}	0.782±0.024	0.566±0.062	0.362±0.092	0.476±0.108	4.25	1.50
	{IA.MR}	0.779±0.020	0.538±0.050	0.321±0.062	0.456±0.097	8.25	2.75
	{IA.TR}	0.772±0.028	0.545±0.111	0.351±0.151	0.411±0.076	9.00	2.25
	{IA}	**0.798±0.034**	0.597±0.043	0.397±0.072	0.493±0.084	**1.75**	**1.00**
ANN	{IA.AMR}	0.787±0.023	0.451±0.066	0.222±0.076	0.468±0.146	8.75	3.75
	{IA.ATR}	0.780±0.020	0.470±0.124	0.261±0.172	0.409±0.144	10.50	3.50
	{IA.MR}	0.777±0.026	0.523±0.201	0.360±0.271	0.412±0.082	8.75	2.25
	{IA.TR}	0.795±0.038	0.512±0.182	0.339±0.227	0.417±0.049	8.00	2.75
	{IA}	0.779±0.037	0.426±0.032	0.198±0.028	0.359±0.107	12.75	4.75
J48	{IA.AMR}	0.740±0.066	0.587±0.099	0.397±0.139	0.459±0.082	6.50	1.50
	{IA.ATR}	0.768±0.019	0.570±0.098	0.378±0.134	0.425±0.070	7.00	1.75
	{IA.MR}	0.746±0.069	**0.599±0.090**	**0.412±0.128**	0.459±0.086	5.00	1.50
	{IA.TR}	0.706±0.062	0.458±0.100	0.244±0.105	0.370±0.091	13.75	4.50
	{IA}	0.757±0.044	0.561±0.103	0.359±0.138	**0.496±0.042**	6.50	2.00
MAX		**0.798±0.034**	**0.599±0.090**	**0.412±0.128**	**0.496±0.042**	**1.75**	**1.00**

ior of the analyzed process: *tandem repeats* (TR), *alphabet tandem repeats* (ATR), *maximal repeats* (MR), and *alphabet maximal repeats* (AMR). When computing the *f-View* representation of a trace, we turned each of these patterns as a non negative integer attribute, storing the number of times the respective pattern occurred in the trace.

Similarly to (Nguyen et al., 2014), we considered the following heterogenous families of patterns: *(i)* {IA}, i.e. individual activities used alone (producing a bag-of-activity representation of traces' structure); *(ii)* {IA, TR}, i.e. the combination of individual activities and of tandem repeats; *(iii)* {IA, ATR}, i.e. individual activities combined with alphabet tandem repeats; *(iv)* {IA, MR}, i.e. individual activities plus maximal repeats; *(v)* {IA, AMR}, i.e. individual activities plus alphabet maximal repeats.

For each pattern family, we used only a selection of those patterns that most frequently occur in the log (namely, 100 patterns of type {IA} and 250 patterns for each other pattern family). In addition, when HO-DDM-mine exploited the re-sampling to mitigate the skewness in the log, all the deviant (i.e. positive) traces in it were duplicated until a deviant-normal ratio of approximately 1:2 is reached.

As to the induction of base DDMs, we resorted to the following methods: the decision-tree learning method *J48* (Quinlan, 1993); the *k-NN* procedure *IBk* (with $k = 10$); the multi-layer perception method (named hereinafter *ANN*) (Zhang, 2000); the *LibSVM* Support-Vector-Machines classifier (Cortes and Vapnik, 1995) with an RDF kernel; and the rule-base classifier *JRip* (Witten and Frank, 2005).

For the induction of a probabilistic combiner model, we tested two alternative settings of algorithm HO-DDM-mine: one using an *AODE*-based classifier, and the other using an *HNB*-based classifier.

5.2 Test Results

A First Look a the Competitor's Results. Since the approach in (Nguyen et al., 2014) consists in applying each learning method to each distinct view of the log (generated according to one of the pattern families described in the previous subsection), it produces the 15 independent DDM models shown in Table 1 —namely, $J48_{\{IA\}}$, ..., $J48_{\{IA+AMR\}}$, $IBk_{\{IA\}}$, ..., $IBk_{\{IA+AMR\}}$, $ANN_{\{IA\}}$, ..., $ANN_{\{IA+AMR\}}$—which should be compared with the ones discovered by our approach. However, it is easy to notice that the outcomes in Table 1 are almost all very close to one another, and no one single DDM can be clearly declared winning over its competitors in all the quality metrics simultaneously. For instance, the *AUC* value for the $IBK_{\{IA\}}$ model is 0.798, while that for $ANN_{\{IA+TR\}}$ is 0.795 —this difference of less than 5% in their values reveals that they are practically equivalents to one another in terms of AUC performances. The same holds for the *G-Mean* of $J48_{\{IA+MR\}}$ (0.599), which is very close again to that of $IBk_{\{IA\}}$ (0.597). Similar considerations can be easily spotted as well for the remaining metrics. In such a situation, choosing the most suitable competitor to run against HO-DDM-mine is not a straightforward task.

Table 2: Prediction results on the $BPIC11_{CC}$ log by HO-DDM-mine and selected configurations of the competitor method (Nguyen et al., 2014). All the values were computed by averaging the results of 5 trials, performed according to a 5 fold cross-validation scheme. For each metrics, the best outcome is reported in bold.

Method	Bayesian Combiner	Setting	AUC	G-Mean	Recall	Precision
Ours	HNB	RES+MORE_LEARNERS	**0.857±0.051**	**0.740±0.018**	**0.606±0.041**	**0.745±0.047**
		RES	0.822±0.040	0.726±0.030	0.592±0.059	0.717±0.044
		NO_RES	0.817±0.024	0.652±0.035	0.477±0.055	0.505±0.077
	AODE	RES+MORE_LEARNERS	0.853±0.053	0.736±0.022	0.598±0.042	0.742±0.049
		RES	0.819±0.044	0.722±0.047	0.584±0.080	0.715±0.047
		NO_RES	0.813±0.026	0.648±0.039	0.469±0.056	0.502±0.082
(Nguyen et al., 2014)	-	BEST_OF_BEST	0.798±0.034	0.599±0.090	0.412±0.128	0.496±0.042
		BEST_AVG_RANK	0.798±0.034	0.597±0.043	0.397±0.072	0.493±0.084
		BEST_AVG_RANK_5%	0.798±0.034	0.597±0.043	0.397±0.072	0.493±0.084

Summarizing the Competitor's Achievements. In order to enable an easier comparison of our approach to the alternative settings of the competitor approach presented above, we devised a method for summarizing the performances of the latter. More precisely, we defined three different criteria for choosing the best achievement of the competitor approach: *(i)* BEST_OF_BEST, *(ii)* BEST_AVG_RANK, and *(iii)* BEST_AVG_RANK_5%.

As to the BEST_OF_BEST row, it simply reports, for each evaluation method, the best value (i.e. the maximum) obtained by all of the different configurations of the approach in (Nguyen et al., 2014) in each single metric. To make clear which values are chosen, the best outcome in each column (i.e. performance metric) of Table 1 have been marked in bold. For the reader' convenience, these values are also explicitly reported in the row with the **MAX** label at the bottom of the same table. Clearly, it is important to point out that this row provides an overestimated evaluation of the competitor approach, which may not correspond to any actual configuration of it. In a sense, this row is a sort of upper bound for the performance of all the considered configurations of the competitor.

Thus, in order to provide a more realistic (yet concise) term of comparison, we defined a second criterion for a further competitor, denoted by BEST_AVG_RANK, aiming at meaningfully aggregating all the results obtained with the approach of (Nguyen et al., 2014) and reported in Table 1. The way this competitor is actually determined is explained in the following.

Let C be the set of all DDM models discovered by the tested methods, and $M = \{AUC, G\text{-}Mean, R, P\}$ be the set of metrics considered in our evaluation setting. For any model $c \in C$ and any metrics $m \in M$, let $score(c,m)$ be the value returned by evaluating m against c. Based on these values, we ranked the models in C over each metrics. More clearly, $rank(c,m) = 1$ (resp. $rank(c,m) = k$) iff c is the best (k-th best) performer according to metrics m. Considering all metrics equally important for assessing the quality of a DDM, we computed an overall average ranking score for each model $c \in C$ as follows:

$$AvgRank(c) = .25 \times (rank(c,AUC) + \\ + rank(c,G-Mean) + rank(c,R) + rank(c,P))$$

The BEST_AVG_RANK model, selected among all the other models discovered by (using different configurations of) the approach of (Nguyen et al., 2014), is the one reaching the highest value of the overall ranking score *AvgRank*.

Example 1. Let us consider the model $IBK_{\{IA\}}$, discovered with method *IBk* on individual-activities features (i.e., by using only the family IA of patterns) According to the values in Table 1, it can be easily noted that $rank(IBK_{\{IA\}}, AUC) = 1$ since $IBK_{\{IA\}}$ is scored higher than any other model on the AUC metric (i.e. it achieved the maximum score over the AUC column). By converse, $rank(IBK_{\{IA\}}, G\text{-}Mean) = 2$, since $IBK_{\{IA\}}$ is the second best performer according to the *G-Mean* metric —the same holds also for the metrics R and P. As a final result, we obtain the overall rank-oriented score $AvgRank(IBK_{\{IA\}}) = .25 \times (1+2+2+2) = 0.25 \times 7 = 1.75$. According to this ranking criterion, the model returned by *IBk* on the IA-based log view is deemed as the best result of the approach in (Nguyen et al., 2014), namely BEST_AVG_RANK, with an average rank of 1.75. □

For the sake of comparison, the row of Table 2 marked as BEST_AVG_RANK reports the quality measures received by this model, as a second term of comparison for our approach.

The way the BEST_AVG_RANK competitor has been computed might be susceptible to criticisms due to numeric approximation problems possibly plaguing very close values. Indeed, it may happen that

Table 3: Prediction results on the $BPIC11_{CC}$ log by HO-DDM-mine (in the configuration RES+MORE_LEARNERS) when using different learning algorithms (as an alternative to our AODE-based and HNB-based Bayesian meta-classifiers) for the discovery of a combiner model. All the values were computed by averaging the results of 5 trials, performed according to a 5 fold cross-validation scheme. For each metrics, the best outcome is reported in bold.

Meta-algorithm	*AUC*	*G-Mean*	*R*	*P*
HNB	**0.857±0.051**	**0.740±0.018**	**0.606±0.041**	**0.745±0.047**
AODE	0.853±0.053	0.736±0.022	0.598±0.042	0.742±0.049
AdaBoostM1	0.811±0.056	0.719±0.039	0.579±0.057	0.710±0.070
J48	0.748±0.057	0.724±0.048	0.584±0.065	0.718±0.071
JRip	0.715±0.029	0.693±0.033	0.543±0.040	0.676±0.061
Logistic	0.789±0.052	0.712±0.033	0.570±0.049	0.696±0.042

two models have performance scores so much close among them (i.e. under a certain approximation threshold z) that could be retained unfair assigning them different ranks. In order to preventively cope with such potential concerns, we considered a third evaluation strategy accounting as equivalent two models $c1, c2 \in C$ w.r.t. a given metric $m \in M$ if the difference between their values falls below a specified threshold z. More formally, this can be stated as in the following:

$$rank(c1,m) = rank(c2,m) \text{ iff}$$
$$|m(c1) - m(c2)| \leq z \times \min(m(c1), m(c2))$$

As a consequence of the definition above, a new rank *AvgRank_z%* can be easily defined. Specifically, in our setting we considered as a reasonable approximation a threshold of 5% (i.e. $z = .05$), and then we selected the competitor BEST_AVG_RANK_5% (cf. last row of Table 2) according to the index *AvgRank_5%*.

Example 2. Let us focus again on the IBk_{IA} model, and let $z = .05$ be the threshold value used for alleviating the numeric approximation problem in our calculus. Based on Table 1, it results that $rank(IBK_{\{IA\}}, AUC) = 1$, as $IBK_{\{IA\}}$ performs better than any other approach over the metric AUC. However, under this new threshold-based setting, $rank(IBK_{\{IA\}}, G\text{-}Mean) = 1$, although the best performer w.r.t. the metric *G-Mean* is $\texttt{J48}_{\{IA,MR\}}$. Indeed, $\texttt{IBK}_{\{IA\}}$ and $\texttt{J48}_{\{IA,MR\}}$ are ranked equally due to the fact that $|G\text{-}Mean(IBK_{\{IA\}}) - G\text{-}Mean(J48_{\{IA,MR\}})| = |0.597 - 0.599| = .002 \leq .05 \times min(0.597, 0.599) = .05 \times 0.597 = .03$. Similar considerations apply for metrics R and P. Therefore, we have that $AvgRank\_5\%(IBK_{\{IA\}}) = .25 \times (1+1+1+1) = 0.25 \times 4 = 1.00$. As a consequence, $\texttt{IBK}_{\{IA\}}$ is the best performer according to criterion *AvgRank_5%*, i.e. the BEST_AVG_RANK_5% model. Please, notice that, by pure chance, it incidentally coincides with the BEST_AVG_RANK model. □

Clearly, by the way it is computed, the performances of our competitor in its BEST_OF_BEST setting are always better than that in both the BEST_AVG_RANK and BEST_AVG_RANK_5% ones. Therefore, the comparative analysis carried out in the following is focused on the ("optimistic" for the competitor) BEST_OF_BEST scenario.

Comparing HO-DDM-mine's Results with the Best Ones of the Competitor. Table 2 reports the results obtained by algorithm HO-DDM-mine, compared with those of the proposed in (Nguyen et al., 2014). For the sake of comparison, we used the same families of patterns and the same (or a slightly wider) set of classification methods as in (Nguyen et al., 2014), and turned the probabilistic classifications of HO-DDM-mine into deterministic ones, using a fixed deviance threshold $\gamma = 0.5$.

Specifically, as far as concerns HO-DDM-mine, we tested three different configurations of it:

- NO_RES, where no resampling procedure is applied to the transformed log (in order to reduce the class imbalance ratio), and the same set of (base) inductive learning methods as in (Nguyen et al., 2014) (namely J48, IBk, ANN) are used;
- RES, using our basic oversampling scheme and the same battery of base classifiers as in the previous configuration (and in (Nguyen et al., 2014));
- RES+MORE_LEARNERS, which uses the same oversampling setting as in configuration RES, while exploiting the classifier-induction methods implemented in our prototype system (i.e. J48, IBk, ANN, LibSVM, JRip).

The three bottommost rows in the table report, as a term of comparison, the best scores obtained by all the settings of the competitor, and computed according to the three criteria (namely, BEST_OF_BEST, BEST_AVG_RANK, and BEST_AVG_RANK_5%) explained in the previous paragraph.

From the figures in Table 2, we can draw several interesting observations. First of all, HO-DDM-mine (irrespective of the kind of probabilistic combiner adopted), even in the basic NO_RES configuration, performs always better (over all the quality metrics) than

the competitor (Nguyen et al., 2014), whatever configuration is used for the latter. This confirms the validity of using an ensemble-learning approach to the deviance detection problem. In particular, it is worth noticing that when allowed to induce an HNB combiner, algorithm HO-DDM-mine achieves better performances than its AODE-based version, despite the latter method make use of an ensemble of SPODEs (see Section 2.1) while HNB only adopts a single Bayesian model. This likely descends from the capability of HNB to effectively capture the inherent dependencies between the attributes in our *s-View* representation.

Performing a finer grain analysis, we can notice that the gain achieved by the HNB-based version of HO-DDM-mine over the approach in (Nguyen et al., 2014) in its basic configuration without resampling becomes even more marked when using a basic oversampling procedure (i.e. in the RES configuration). Specifically, even though the increment in terms of *AUC* is moderate (3.00%), we can observe a significant improvement for the metric *G-Mean* (21.20%), and a noticeable 44.56% (resp. 43.69%) achievement in terms of precision (resp. recall).

Further improvement is obtained by our approach (still equipped with HNB) when letting it use both the oversampling procedure and a broader range of base classifiers (i.e. configuration RES+MORE_LEARNERS) —actually, we only extended the learning methods used by our competitor with the insertion of *LibSVM* and *JRip*. Indeed, in this case, a gain of 7.39% (resp. 23.54%, 50.20%, 47.09%) is obtained in terms of *AUC* (resp. *G-Mean*, precision, recall) w.r.t. the overestimated BEST_OF_BEST configuration.

Notably, a similar trend in the results can be found for the version of HO-DDM-mine adopting AODE as combiner, and then it does not require any further supplementary discussion.

In summary, it seems that the combination of an oversampling method with our ensemble-learning strategy helps obtain higher improvements (w.r.t. the competitor supervised deviance-detection approach) than exploiting a wider range of base classifiers.

Benefits of using Bayesian Combiners. In a further series of tests, we considered some variants of algorithm HO-DDM-mine where we tried the following meta-learning methods (for inducing the combiner model $\hat{c}$ of any HO-DDM) as an alternative to its native Bayesian classifiers: AdaBoostM1, J48, JRip, Logistic. The results of this experimentation are reported in Table 3. It is clear that both kinds of Bayesian combiner allow to achieve superior performances, over all the quality metrics, than these alternative methods. This is likely due to their capability of obtaining accurate and robust estimates of the class membership probability, despite the high degree of dependence between the attributes in the stacked view given as input to it.

6 CONCLUSION

In this paper, we have extended a previous proposal where an innovative ensemble-learning framework for mining business process deviances that exploits multi-view learning has been provided. Relevant contributions, which clearly confirm the flexibility, the reliability and the effectiveness of the general deviance detection framework, respectively, have been introduced and experimentally assessed via a wide and comprehensive experimental campaign. Notice that our multi-view learning approach is neatly different from many others in the literature that do not follow an ensemble-based scheme (such as, e.g., (Blum and Mitchell, 1998; Nigam and Ghani, 2000; Wang and Zhou, 2010)), and somewhat rely on the fact that the different data views are (more or less) conditionally independent of one another given the label —an assumption that does not hold in the setting considered in this work, where the views are likely to feature high levels of redundancy and of inter-dependence.

As future work, we mainly plan to investigate on the combination of our ensemble learning approach with log-oriented clustering methods (like those in (Folino et al., 2014; Bose and van der Aalst, 2010)), as well as to apply the framework to real-life projects, in order to derive new test-beds, new requirements, and new solutions. Also, to provide effectiveness, we aim at studying the integration of intelligent data processing techniques (e.g., (Cuzzocrea and Matrangolo, 2004; Cuzzocrea, 2006; Cuzzocrea et al., 2009)).

REFERENCES

Bose, R.P.J.C., van der Aalst, W.M.P.: Discovering signature patterns from event logs. In: IEEE Symp. on Computational Intelligence and Data Mining (CIDM'13), pp. 111–118 (2013)

Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)

Buckland, M., Gey, F.: The relationship between recall and precision. Journal of the American Society for Information Science 45(1), pp. 12–19 (1994)

Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), pp. 273–297 (1995)

Cuzzocrea, A.: Accuracy Control in Compressed Multidimensional Data Cubes for Quality of Answer-based OLAP Tools. In: Proc. of IEEE SSDBM 2006, pp. 301–310, 2006

Cuzzocrea, A., Folino, F., Guarascio, M., Pontieri, L.: A Multi-view Learning Approach to the Discovery of Deviant Process Instances. In: Proc. of CoopIS 2015, pp. 146–165, 2015

Cuzzocrea, A., Furfaro, F., Saccà, D.: Enabling OLAP in mobile environments via intelligent data cube compression techniques. Journal of Intelligent Information Systems (33)(2), pp. 95–143 (2009)

Cuzzocrea, A., Matrangolo, U.: Analytical Synopses for Approximate Query Answering in OLAP Environments. In: Proc. of DEXA 2004, pp. 359–370, 2004

van Dongen, B.: http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

van Dongen et al.: The ProM framework: A new era in process mining tool support. In: Proc. of 26th 10th Int. Conf. on Applications and Theory of Petri Nets (ICATPN'05), pp. 444–454 (2005)

Frank, E., Hall, M.A., Holmes, G., Kirkby, R., Pfahringer, B.: Weka - a machine learning workbench for data mining. In: The Data Mining and Knowledge Discovery Handbook, pp. 1305–1314 (2005)

Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intelligent Data Analysis 6(5), pp. 429–449 (2002)

Kubat, M., Holte, R., Matwin, S.: Learning when negative examples abound. In: Proc. of 9th Europ. Conf. on Machine Learning (ECML'97), pp. 146–153 (1997)

Langley P., Iba W., Thompson K.: An analysis of Bayesian classifiers. In: Proc. of 10th Nat. Conf. on Artificial intelligence (AAAI'92), pp. 223–228 (1992)

Lo, D., Cheng, H., Han, J., Khoo, S.C., Sun, C.: Classification of software behaviors for failure detection: A discriminative pattern mining approach. In: Proc. of 15th Int. Conf. on Knowledge Discovery and Data Mining (KDD'09), pp. 557–566 (2009)

Nguyen, H., Dumas, M., Rosa, M.L., Maggi, F.M., Suriadi, S.: Mining business process deviance: A quest for accuracy. In: Proc. of 2014 Int. Conf. On the Move to Meaningful Internet Systems (OTM'14), pp. 436–445 (2014)

Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)

Folino, F., Guarascio, M., Pontieri, L.: Mining predictive process models out of low-level multidimensional logs. In: Proc. of 26th Int. Conf. on Advanced Information Systems Engineering (CAISE'14), pp. 533–547 (2014)

Blum A. and Mitchell T.: Combining labeled and unlabeled data with co-training. In: Proc. of the 11th Conf. on Computational Learning Theory (COLT'98), pp. 92–100 (1998)

Nigam K., Ghani R.: Analyzing the effectiveness and applicability of co-training. In: Proc. of the 9th Int. Conf. on Information and Knowledge Management (CIKM'00), pp. 86–93 (2000)

Wang W., Zhou Z.H.: A new analysis of co-training. In: Proc. of the 27th Int. Conf. on Machine Learning (ICML'10), pages 1135–1142, 2010.

Domingos P., Pazzani M.J.: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In: Proc. 13th Int. Conf. on Machine Learning (ICML'96). pp.105–112 (1996)

Domingos P., Pazzani M.J.: On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. Machine Learning 29, pp.103–130 (1997)

Keogh E.J., Pazzani M.J.: Learning the Structure of Augmented Bayesian Classifiers. Int. J. Artificial Intelligence Tools, 11(40), pp. 587–601 (2002)

Ying Y. et al.: To Select or To Weigh: A Comparative Study of Linear Combination Schemes for SuperParent-One-Dependence Estimators. IEEE Transactions on Knowledge and Data Engineering, 19(12), pp.1652–1665 (2007)

Bose, R.P.J.C., van der Aalst, W.M.P.: Trace clustering based on conserved patterns: Towards achieving better process models. In: Proc. of Business Process Management Workshops (BPI'10), vol. 43, pp. 170–181 (2010)

Sahami M.: Learning Limited Dependence Bayesian Classifiers. In: Proc. 2nd ACM SIGKDD of Int. Conf. Knowledge Discovery and Data Mining (KDD'96), pp. 334–338 (1996)

Suriadi S., Chun O., van der Aalst W.M.P., ter Hofstede A.H.M. : Root Cause Analysis with Enriched Process Logs. In: Business Process Management Workshops 2012, pages 174–186, 2013.

Swinnen, J., Depaire, B., Jans, M.J., Vanhoof, K.: A process deviation analysis - A case study. In: Proc. of 2011 Business Process Management Workshops, pp. 87–98 (2011)

Webb G.I., Boughton J., Wang Z. Not So Naive Bayes: Aggregating One-Dependence Estimators. Machine Learning, 58(1), pp. 5–24 (2005)

Zhang, G.P.: Neural networks for classification: a survey. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 30(4), pp. 451–462 (2000)

Zhang, H., Jiang, L., Su, J.: Hidden naive bayes. In: Proc of AAAI, pp. 919–924 (2005)

Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc. (2005)