# Run-time prediction of business process indicators using evolutionary decision rules

Alfonso E. Márquez-Chamorro, Manuel Resinas, Antonio Ruiz-Cortés, Miguel Toro

*Dpto. Lenguajes y Sistemas Informáticos, University of Seville, Seville, Spain*

*Keywords:*
Business process management
Process mining
Predictive monitoring
Business process indicator
Evolutionary algorithm

A B S T R A C T

Predictive monitoring of business processes is a challenging topic of process mining which is concerned with the prediction of process indicators of running process instances. The main value of predictive monitoring is to provide information in order to take proactive and corrective actions to improve process performance and mitigate risks in real time. In this paper, we present an approach for predictive monitoring based on the use of evolutionary algorithms. Our method provides a novel event window-based encoding and generates a set of decision rules for the run-time prediction of process indicators according to event log properties. These rules can be interpreted by users to extract further insight of the business processes while keeping a high level of accuracy. Furthermore, a full software stack consisting of a tool to support the training phase and a framework that enables the integration of run-time predictions with business process management systems, has been developed. Obtained results show the validity of our proposal for two large real-life datasets: BPI Challenge 2013 and IT Department of Andalusian Health Service (SAS).

## 1. Introduction

Process mining techniques allow the extraction of useful information from the event log and historical data of business processes (Li et al., 2016a; Schnig, Cabanillas, Jablonski, & Mendling, 2016). Knowledge can be generated from this information to improve the processes (Kamsu-Foguem, Rigal, & Mauget, 2013; Nkambou, Fournier-Viger, & Mephu-Nguifo, 2011; Potes-Ruiz, Kamsu-Foguem, & Grabot, 2014). Generally, this knowledge is extracted after the process has been finished. Nevertheless, the interest to apply process mining to running process instances is increasing ( Maggi, Francescomarino, Dumas, & Ghidini, 2014). One of the main issues in process mining is the predictive monitoring of business processes (de Leoni, van der Aalst, & Dees, 2016). The main value of predictive monitoring is to provide information in order to take proactive and corrective actions to improve process performance and mitigate risks in real time. Predictive monitoring of business process provides the prediction of business process indicators of a running process instance with the generation of predictive models. Business process indicators are quantifiable metrics that can be measured directly by data that is generated within the process flow (del Río-Ortega, Resinas, Cabanillas, & Ruiz-Cortés, 2013). An improvement in the prediction of these indicators, in many occasions, also means savings in human and economic resources and prevention of important loss of turnover to the companies. Some issues of real companies can also be solved with predictive monitoring. For instance, Push to front problem, detailed in Verbeek (2013) and covered in this work, try to identify those incidences which are not resolved by the service desks and are pushed to the other support lines of the company.

Since predicting these process indicators can be interpreted as a classification or regression problem, machine learning algorithms can be used for this task (Francescomarino, Dumas, Maggi, & Teinemaa, 2015; Maggi, Francescomarino, Dumas, & Ghidini, 2014). Classification and regression are used for the prediction of discrete or continuous target values, respectively. For instance, an indicator such as, the cycle time of a process instance can be regarded as a regression problem. By contrast, the fulfillment of a determined target, *e.g.* the process instance must complete in less than 4 h, or a condition, *e.g.* whether a specific activity occurs in the process instance, can be interpreted as a classification problem.

Multiple machine learning approaches have been applied for predictive monitoring, such as decision trees (Maggi, Francescomarino, Dumas, & Ghidini, 2014), clustering methods (Francescomarino, Dumas, Maggi, & Teinemaa, 2015) or neural networks (Tax, Verenich, Rosa, & Dumas, 2016). Nevertheless, as far as we are concerned, evolutionary algorithms (EAs) have not been applied for the prediction of process indicators. The use of an evolutionary algorithm may be justified for four different reasons (Fogel, 1997): *(a)* it can handle continuous and discrete attributes and automatically discretizes the continuous features; *(b)* it also handle missing attribute values and noise; *(c)* it can build models that can be easily interpreted by humans and finally *(d)* it finds a sub-set of the features that are relevant to the classification without the use of feature selection. In addition, EAs have shown the capacity of finding suboptimal solutions in search spaces when the search space is characterized by high dimensionality (Marquez-Chamorro, Asencio-Cortes, Divina, & Aguilar-Ruiz, 2014). In this case, the set of possible state conditions of a process, encoding in decision rules, determine the search space and fulfil these requirements. Some methods in process mining area also utilize association or decision rules for the improvement of the performance of the processes (Karray, Chebel-Morello, & Zerhouni, 2014; Wen, Zhong, & Wang, 2015).

In this work, we have developed a general method based on an evolutionary rule learning approach for the prediction of business process indicators in execution time. The resulting model consists in a set of decision rules that determine a prediction for an indicator of a running process instance. We have employed as encoded features, a window of the previous events to the point in the process execution where the prediction is carried out. This window of events considers attributes of a typical event log, such as activity name or timestamps, together with the data of each event. A combination of continuous and discrete values is allowed by the evolutionary algorithm. An advantage of this approach is that the generated decision rules can be interpreted by users to extract further insight of the business processes. Furthermore, as previously mentioned, the method incorporates a new encoding based on event windows of different sizes which provides more information from event logs. Additionally, this method is accompanied by a full software stack we have developed to support both the training and the prediction phase of our predictive monitoring approach. The learning phase is supported by a ProM plugin that helps in the computation of process indicators and the preprocessing of the event log for the machine learning algorithm. The prediction phase is supported by a framework that enables the integration of run-time predictions obtained from the predictive models generated by the training phase with business process management systems like Camunda (Camunda, 2016).

Our approach was exhaustively tested with two different real-life event logs to assess the validity of the proposal. The datasets belong to IT Department of Health Services of Andalusia (Spain) and the BPI 2013 Challenge (Verbeek, 2013). For the validation of the proposal, we also include a comparison with a method of the literature, described in Breuker, Matzner, Delfmann, and Becker (2016), and several machine learning approaches, under the same experimental conditions, in order to justify the use of the evolutionary algorithm.

The remainder of this paper is organized as follows. Section 2 introduces the main concepts referred throughout the paper. Section 3 summarizes the related work in this area. Section 4 introduces our methodology. Section 5 presents the experimentation and obtained results. Finally, Section 6, includes some conclusions and possible future works.

## 2. Background on predictive monitoring

The goal of predictive monitoring is to predict some aspect of the execution of a running process instance. To do so, it relies on the existence of an event log that contains the relevant information of the execution of a business process. An event log ($L$) is composed of a set of traces ($T$) that contain each event ($E$) that occurs in the different instances of a business process. Each execution of a process instance is reflected in a trace. Formally, we can express a trace $T_i$ as a list of events $T_i = [E_{i_1}, ..., E_{i_m}]$ where $E_{i_1}$ represents the first event and $E_{i_m}$ reflects the final event of the execution of the process instance. An event ($E$) represents an instant of the execution of an activity of the process. Each event contains a set of attributes or properties ($p$) which represents all the information for the definition of such event, *e.g.* timestamp or the resource that execute a determined activity $E_j = [p_{j_1}, ..., p_{j_n}]$ where $n$ determines the total number of properties of the event. Finally, we can represent a log as a sequence of instances which have finished in an interval of time $L = [T_1, ..., T_m]$ where $T_1$ represents the first executed trace and $T_m$ is the last execution trace in the time interval.

**Table 1**
Event log example.

| case id | event id | timestamp | activity | resource | cost |
|---|---|---|---|---|---|
| 1 | 107561 | 12-12-2016:12.15 | A | Lucas | 100 |
| | 107562 | 12-12-2016:14.55 | B | Lucas | 300 |
| | 107563 | 12-12-2016:17.30 | C | Paul | 200 |
| | 107564 | 13-12-2016:12.15 | D | Laura | 400 |
| 2 | 108631 | 14-12-2016:10.00 | A | Fred | 100 |
| | 108632 | 14-12-2016:12.52 | D | Fred | 200 |
| | 108633 | 14-12-2016:13.27 | E | Barney | 100 |
| 3 | 108945 | 15-12-2016:10.32 | B | Alan | 100 |
| | 108946 | 16-12-2016:09.18 | E | Sylvia | 300 |

Table 1 depicts an example of event log. Each grouped row represents a trace. Therefore, the example shows 3 traces (case id: 1, 2 and 3) consisting of 4, 3 and 2 events, respectively (event id row). The number of events of each trace can be different. Finally, each event has several properties. In the table, four of them are depicted, namely: *timestamp, activity, resource* and *cost*.

A business process indicator ($I$) is a quantifiable metric that can be measured directly by data that is generated within the process flow. There are two types of process indicators: instance-level indicator and aggregated indicator (del Río-Ortega, Resinas, Cabanillas, & Ruiz-Cortés, 2013). An instance-level indicator provides a metric for a single process instance. It can be defined as a function of a trace $T$, *i.e.* $I(T)$, which is calculated by using the values of the attributes of the events that belong to this trace. This function can return a Boolean value, *e.g.* a determined condition fulfilled by the trace, or a real value, *e.g.* the duration of an activity. An aggregated indicator ($I_A$) can be represented as a function $I_A(L^f)$, where $L^f = \{T_i \in L \mid filter(T_i)\}$ contains all the traces $T_i$ from an event log $L$ that fulfill a determined requirement $filter(T_i)$. Generally, this filter is defined as a temporal constraint, such as an interval of time. In this work, we consider only instance-level indicators.

The goal of predictive monitoring is to predict the value of the indicator before a process instance finishes by means of a predictive model, which are usually built based on the information provided by the traces of previous process instances. Therefore, a predictive model for an indicator $I$ is a function $P_I([E_{i_k}, \ldots, E_{i_l}])$, with $k \leq l$, that computes a prediction for $I$ from the partial trace $[E_{i_k}, \ldots, E_{i_l}]$, where $E_{i_l}$ is the last event that have occurred in trace $T_i$ at a given moment. If $k = 1$, then all events that have occurred in the process instance at hand are considered. Otherwise, if $k = l$, then only the last event of the process instance is considered.

There are two types of predictions that are usually relevant in the context of predictive monitoring, namely: next-event prediction and end-of-instance prediction. In next-event prediction, the goal is to predict the value of the indicator for the next event in the process instance ($E_{i_{l+1}}$), *i.e.*, $P_I([E_{i_k}, \ldots, E_{i_l}])$ is approximately $I([E_{i_1}, \ldots, E_{i_{l+1}}])$. On the other hand, in end-of-instance prediction, the goal is to predict the value of the indicator at the end of the process instance, *i.e.*, $P_I([E_{i_k}, \ldots, E_{i_l}])$ is approximately $I([E_{i_1}, \ldots, E_{i_m}])$, where $E_{i_m}$ is the last event of the process instance.

The prediction of one indicator or the other usually depends on the type of indicator and the moment in which actions to mitigate the risks of not fulfilling an indicator can be put in place. In this work, we address both types of predictions.

## 3. Related work

Several approaches for the predictive monitoring of business process indicators can be found in the literature. We have classified some of these methods according to the type of predicted outcome, such as time, risk indicators, SLA violation indicators and other indicators.

Time is one of the most valuable indicators during the execution of a business process. Following two works predict the expected time of the process. In Polato, Sperduti, Burattin, and de Leoni (2014), authors present a Naive–Bayes (NB) and (support vector regression) SVR approach to predict the remaining time of a running process which takes into account two different factors: the likelihood of the future states of a transition system calculated by a NB technique and the remaining time of the process given by a regression model. The work described in Rogge-Solti and Weske (2015), presents a time prediction method based on non-Markovian Petri net enriched with duration distributions. The method predicts the expected remaining duration of a running process instance and also predicts the risk of reaching a temporal deadline.

Following works consider risk factors across all running process instances for the prediction. The method proposed in Conforti, de Leoni, Rosa, van der Aalst, and ter Hofstede (2015) measures the likelihood and severity of a fault in the system, providing a decision tree based technique for the risk predictions. In Pika, van der Aalst, Fidge, ter Hofstede, and Wynn (2013), authors propose a set of process risk indicators (5 different types) than can be predicted by using a statistical method. This method, based on the detection of outliers, uses standard deviations to determine if a value follow a normal distribution. Prediction of risks is taken into account in the work presented in Conforti, de Leoni, Rosa, and van der Aalst (2013). The risks are predicted using decision trees generated from the event logs. Attributes such as process data, involved resources, task durations and other contextual information were considered by the algorithm. Authors describe in Pika, van der Aalst, Wynn, Fidge, and ter Hofstede (2016) a method for identifying risk factor in business processes. They calculate current aggregate values for the different risks and estimates a prediction from an annotated Petri net.

The prediction of SLA violations is also taken into account in several proposals. The clustering-oriented method, presented in Folino, Guarascio, and Pontieri (2012) predicts processing times and associated SLA violations. The instance is assigned to a reference scenario (cluster) which is used for the prediction. The predictive model is based on decision trees and is called Predictive Clustering Tree (PCT). In Francescomarino, Dumas, Maggi, and Teinemaa (2015), authors describe a clustering method for the prediction of the fulfilment or violation of a determined service level objective (SLO). A cluster is assigned for each running instance and decision tree is built for each cluster to determine the prediction. In Leitner, Hummer, and Dustdar (2013), authors present the PREVENT tool (prediction and prevention based on event monitoring), which uses multilayer artificial neural networks for the prediction of quantitative SLOs and C4.5 decision trees for qualitative SLOs.

Other process indicators, such as an abnormal termination of the process, are predicted in the following proposals. In Cabanillas, Ciccio, Mendling, and Baumgrass (2014), authors define the monitoring of tasks as a set of requirements for a predictive system. A support vector machine (SVM) approach is used to classify whether the execution will lead to a successful completion or not according to the event log data. Authors describe in Kang, Kim, and Kang (2012) a real-time monitoring system which predicts the abnormal termination of a running business process. The machine learning approach employed is a KNN technique in combination with a local outlier factor (LOF) approach as a fault detection algorithm. A general framework for the prediction is presented in de Leoni, van der Aalst, and Dees (2016). This work analyzes the correlation among the different variables of the event log and constructs a classification or regression tree for the prediction of a problem. Clustering is also applied to identify traces with similar behavior.

According to the predicted outcomes in the literature, the types of predicted indicator can be an estimation of the value of a basic indicator (continuous, *e.g.* the remaining execution time, or discrete, *e.g.* an incident is reopened or not), an aggregate attribute, which is an indicator calculated in an interval of time (*e.g.* the proportion of incidents solved in a month), the probability that a determined risk occurs in the process (*e.g.* the violation of a constraint), a logical predicate, such as a LTL formula, which determines that a certain situation in the process occurs or the prediction of the following event of the running process instance.

A difference between some works in the literature, *e.g.* Polato, Sperduti, Burattin, and de Leoni (2014), Rogge-Solti and Weske (2015) and Pika, van der Aalst, Wynn, Fidge, and ter Hofstede (2016), with respect to our approach is that they only receive as input a process model with the sequence of events with the consequent loss of information. Nevertheless, our approach considers the event data of the activities to perform the prediction using an event window-based encoding which provides enough information and reduces the computational cost. Furthermore, some of the works utilize a synthetic database, *e.g.* Pika, van der Aalst, Wynn, Fidge, and ter Hofstede (2016). In our work, all the experimentation was performed using two real-life datasets. Moreover, we provide comprehensible models based on decision rules to extract further insights of the business processes. Instead, some of the predictive models provided by the majority of methods, such as SVM (Cabanillas, Ciccio, Mendling, & Baumgrass, 2014), SVR (Polato, Sperduti, Burattin, & de Leoni, 2014) or Neural networks (Leitner, Hummer, & Dustdar, 2013), are hardly understandable by humans without a previous preprocessing. In addition, the lack of comparison among proposals in most of the cited works is evidenced, *e.g.* Cabanillas, Ciccio, Mendling, and Baumgrass (2014), Rogge-Solti and Weske (2015), Pika, van der Aalst, Wynn, Fidge, and ter Hofstede (2016) and de Leoni, van der Aalst, and Dees (2016). This is due to the lack of available software and datasets which makes difficult a reliable comparison. We have determined that only three proposals are publicly available but the characteristics of these works are not appropriated for a real comparison with our proposal. Thus, we provide the source code and all the necessary datasets (under demand) for the replication of the experimental study described in this paper. All this information is available in our web page (Marquez-Chamorro & Resinas, 2017). We have also provided a comparison with several machine learning approaches, obtaining a good performance in terms of precision, recall and specificity under the same experimental conditions and have also included a comparative analysis with a recent work of the literature, which justify the use of the evolutionary algorithm. Finally, our approach introduces a full software stack to support all the stages of the predictive monitoring process (preprocessing, training and prediction) and can be integrated with a business process management system like Camunda (Camunda, 2016). On the other hand, other works, *e.g.* Folino, Guarascio, and Pontieri (2012), Francescomarino, Dumas, Maggi, and Teinemaa (2015), Kang, Kim, and Kang (2012) and Conforti, de Leoni, Rosa, van der Aalst, and ter Hofstede (2015), only consider the prediction stage in their software tools.

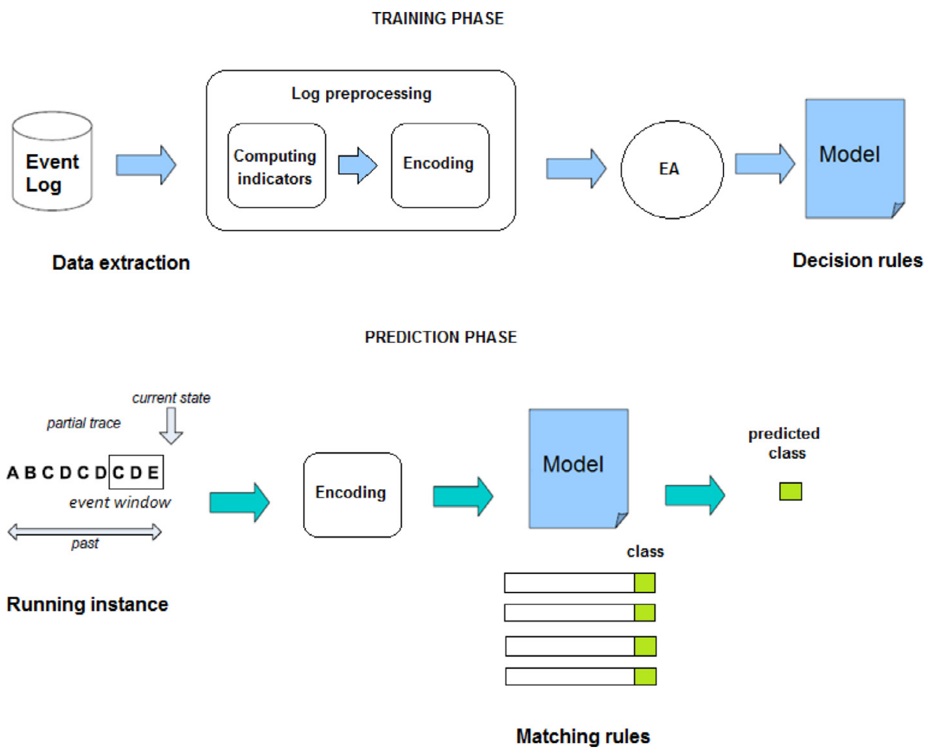


**Fig. 1.** Procedure scheme.

## 4. Predictive monitoring with evolutionary algorithms

This section describes our proposal for the prediction of business process indicators. In particular, our proposal is based on an evolutionary algorithm (EA) using the information data of a window of events described in the event logs. This section is divided into three different subsections. The first one introduces the procedure of predictive monitoring. The calculation of indicators and the encoding of the algorithm are detailed in Section 4.2. Finally, the evolutionary method and a brief introduction of EAs is described in Section 4.3. The evolutionary operators, fitness function and settings of the algorithm are also detailed in this section.

### *4.1. Procedure*

Fig. 1 represents the procedure adopted in our work. This procedure in divided into two phases. First, in the training phase, the event log of a process is obtained and preprocessed. These preprocessing includes the calculation of the process indicators to be predicted and the encoding of the event traces of the log in feature vectors that can be interpreted by the classifiers. These feature vectors are composed by the properties of the different events. In addition, for each feature vector, a class that corresponds to the value of the indicator which we are aimed to predict, is assigned. The preprocessing stage may also include other steps that are common before the application of machine learning algorithms such as the use of techniques to address the imbalanced class problem (Branco, Torgo, & Ribeiro, 2016). Once the data is preprocessed, these feature vectors will constitute the training dataset for the machine learning algorithm. Then, our evolutionary algorithm is trained and generates, as predictive model, a set of decision rules as output of the training phase.

During the prediction phase, the model generated in the training phase can be used to predict at runtime the outcome of running process instances. The input to this phase is the partial log of a running process instance. This partial log is first encoded following the same approach as in the training phase in order to obtain a feature vector. Then, the learned predictive model is applied to the feature vector to determine the predicted value for the process indicator.

### *4.2. Log preprocessing: computing indicators and encoding*

The goal of the preprocessing is to transform the event log into a format that can be used as input for machine learning algorithms. Most machine learning algorithms receive as input a set of feature vectors that contains a target attribute with the value to be predicted. These set of feature vectors conform the training set of the algorithm.

To build the feature vector two steps are required. First, it is necessary to compute the value of the indicator to be predicted. This value is computed according to the existing attributes of the trace, as a result of a combination or arithmetic operation between two or more properties (*e.g.* the sum of all the activity durations).

Second, it is necessary to convert the event log into feature vectors. There are several possible ways to do so. In our case, we follow an approach based of windows of events. Our system parses all the traces of an instance from the event log. For each instance, we represent all the possible combinations of consecutive events and their properties for a determined window size. Table 2 show a representation of an encoding example from the event log depicted in Fig. 1. In this case, the size of the event window is 3. Therefore, the vectors consist of the properties of the events A, B, C and B, C, D for the case 1. These properties are *eventid* (EI), *timestamp* (TS), *resource* (RS) and *cost* (C). Finally, the class represents the value of a process indicator, and is assigned at the end of each vector. For the next-event prediction the class represents a value of the indicator when the following event occurs. For the end-of-instance prediction, the class is calculated according to the value of the indicator at the end of the execution.

Formally, we define the feature vector $v$, as a window of events of size $N$. The information contained for each event property is in-

**Table 2**
Encoded vectors from the event log example showed in Fig. 1.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 ***ABC*** | $EI_A$ | $EI_B$ | $EI_C$ | $TS_A$ | $TS_B$ | $TS_C$ | $RS_A$ | $RS_B$ | $RS_C$ | $C_A$ | $C_B$ | $C_C$ | ... | class |
| 1 ***BCD*** | $EI_B$ | $EI_C$ | $EI_D$ | $TS_B$ | $TS_C$ | $TS_D$ | $RS_B$ | $RS_C$ | $RS_D$ | $C_B$ | $C_C$ | $C_D$ | ... | class |
| 2 ***ACB*** | $EI_A$ | $EI_C$ | $EI_B$ | $TS_A$ | $TS_C$ | $TS_B$ | $RS_A$ | $RS_C$ | $RS_B$ | $C_A$ | $C_C$ | $C_B$ | ... | class |
| 2 ***CBD*** | $EI_C$ | $EI_B$ | $EI_D$ | $TS_C$ | $TS_B$ | $TS_D$ | $RS_C$ | $RS_B$ | $RS_D$ | $C_C$ | $C_B$ | $C_D$ | ... | class |

dicated as $p_{i,j}$ such that $1 \le j \le L$ where $L$ is the maximum number of properties of an event, and $1 \le i \le N$, as it can be shown in Eq. (1) for $N = 3$. Last feature corresponds to the class which indicates a value of the indicator to be predicted. Each attribute can be nominal or a real number.

$$v = [p_{1,1}, p_{2,1}, p_{3,1}, p_{1,2}, p_{2,2}, p_{3,2}, \ldots, p_{1,j}, p_{2,j}, p_{3,j}, class] \quad (1)$$

The size of the event window has a direct impact on the effectiveness and efficiency (computational and time cost) of the execution of the algorithm. Therefore, the selection of this parameter is an important issue which is considered in the Experimentation section.

These feature vectors will determine the set of decision rules generated by our evolutionary algorithm at the end of its execution. This set of rules will be considered our predictive model. We can define a decision rule as a subset of properties $p$ of $N$ events, where $N$ is the size of the event window. An example of one of the decision rules for a subsequence of events A, B and C of the log depicted in Fig. 1 is showed in the following:

$$\begin{aligned}&if\ EI_A \in [107,000, 108,000]\ and\ TS_A = [12-12-2016 : 12.00]\\ &and\ RS_B = [Fred, Laura]\ and\ C_A \in [200, 400]\ and\ C_C \in [300, 400]\\ &and\ RS_C = [Fred]\ then\ true\\ &Accuracy = 0.885 \qquad (2)\end{aligned}$$

This rule determines a true value of a generic indicator, when all the antecedents of the rule are fulfilled. Note that only the representative properties appear in the rule and their values are presented in intervals.

### 4.3. Evolutionary algorithm

Although evolutionary algorithms has been applied to several issues in process mining, such as process model discovery (Bratosin, Sidorova, & van der Aalst, 2010; Medeiros, Weijters, & van der Aalst, 2007), they have never been applied to the runtime prediction of process indicators.

Evolutionary algorithms are population-based stochastic iterative optimization techniques based on the Darwinian concepts of evolution. EA tackles difficult problems by evolving approximate solutions of an optimization problem. An application of EAs is the genetic algorithm (GA). The mechanisms used by GAs to carry out this search can be seen as a metaphor of the processes of biological evolution (reproduction and mutation). This kind of optimization and search algorithms can be applied to solve optimization problems in various fields.

A general scheme of a GA starts with initialization. In this step, an initial population is randomly generated. This population consists of a set of individuals which represent possible solutions of the problem. In our problem, each individual represents a window of $N$ events of a trace, and is composed of a set of event properties or attributes and a class value which reflects the value of a determined process indicator, as we have described in Section 4.2. Therefore, each individual describes a set of conditions to determine a value of a process indicator. In our EA, each one of the attributes are defined using a lower and upper bound. Thus, EA converts nominal-coded attributes in real-coded attributes to facilitate the calculation of the evolutionary operators.

The second step is the evaluation of the individuals that will apply a fitness function to know how “good” is the encoded solution. In this case, the fitness of a particular individual corresponds to the accuracy or the percentage of test instances correctly classified by the rule.

After this, the algorithm performs a number of generations. The GA should stop when the optimal solution is reached, or other stopping criteria is fulfilled. Typically two criteria are established: running a maximum number of generations or the algorithm stops when there are not changes in the population.

Until the stop condition is not fulfilled, the algorithm performs the following steps. A selection operator selects a number of individuals according to the fitness of the individuals, where fitter individuals have more chances of being selected, simulating the concept of survival of the fittest. Our method randomly choose two individuals among all the chromosomes of the population.

Offsprings are generated with the application of crossover and mutation. The crossover and mutation operator are applied according to a given probability. For the crossover, a determined number of positions of the chromosome of the parent individuals are randomly selected. Then, the genes are exchanged on both sides of these positions and the new descendants are generated. A simple mutation randomly replaces a selected attribute value with an allowed value between a determined range. Offsprings are then evaluated and a new population is created. The best individual is selected and added to the population of the next generation (elitism).

At the end of the execution of the algorithm, the best individual is extracted as a solution of the proposed problem. This individual constitutes one of the resulting rules of the set of solutions of the problem. When a rule is generated by our EA, all the training examples covered by that rule are deleted. Therefore, repeated rules are not generated by the algorithm.

The settings of our EA and fitness function are explained in the following. The algorithm starts with a randomly initialized population of 100 individuals and is run for a maximum number of generations (100). In order to obtain the next generation, individuals are selected with a tournament selection mechanism of size two. Crossover and mutation are then applied in order to generate offsprings. We have used a 1-point crossover operator and two different mutation operators, creep and simple random mutation, as explained in Corcoran and Sen (1994). The crossover and mutation probabilities were set to 0.8 and 0.5 respectively. After having performed several runs of the algorithm, the best results were obtained for the cited parameter settings. The fitness function consists in calculating the number of examples correctly classified by the rule, *i.e.* $fitness = P/P_t$ where $P$ is the number of correctly predicted examples and $P_t$ represents the total number of examples. The best individual of the population is maintained in the next generation of EA, following an elitist scheme. The obtained rules generated after each execution of the algorithm, constitute the solution set of rules which are part of the predictive model.

## 5. A software system for predictive monitoring

Two software systems were developed for our proposal: one of them is used to preprocess the event log during the training phase, whereas the other is used to support the prediction phase. In addition, a generic machine learning software was used to support

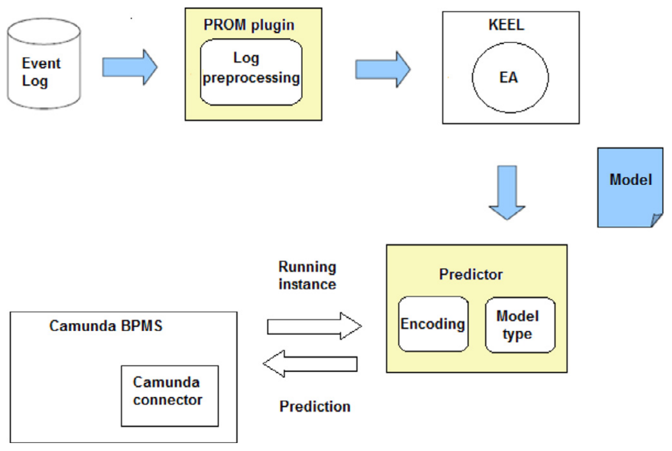


**Fig. 2.** Software systems architecture.

the process of creating the prediction model. The reason why we decided not to include support for building the prediction model together with the log preprocessor is because there are dozens of machine learning frameworks that provide very useful tools to build and evaluate prediction models, so we wanted to left this open so that each user can choose the framework they find more useful in each case. In this paper, we have used KEEL (Alcala-Fdez et al., 2011), which is a framework specially targeted to evolutionary algorithms. Fig. 2 depicts how these tools can be used together. The colored components of the figure were implemented by us. Next we detail each of them.

### 5.1. *Log preprocessor*

This software component has been developed to support the task of preprocessing the log, which includes indicators computation and log encoding, in order to obtain a set of feature vectors that is used as input of the machine learning algorithm. The software has been developed in Java as a plugin (Marquez-Chamorro, 2016) of the ProM framework (van Dongen, de Medeiros, Verbeek, Weijters, & van der Aalst., 2005), which is the most used framework for process mining. The inputs of the plugin, showed in Fig. 3, are three, namely: the log event in XES format (a XML-based standard for event logs), the specifications for the calculation of the indicator in JSON format (JavaScript Object Notation) and the configuration of the encoding, which, for the encoding described in Section 4.2 includes the specification of the window size. In the current version only this encoding is implemented. However, new encodings can be easily added to the system. The output of the plugin is a set of encoded feature vectors that can be directly used as an input for any machine learning algorithm.

Internally, the plugin uses the PPINOT Tool Suite (del Rio-Ortega, 2017) described in del Río-Ortega et al. (2017), to compute the indicator. Therefore the JSON format that specifies how the indicator is computed is defined following the PPINOT metamodel described in del Río-Ortega, Resinas, Cabanillas, and Ruiz-Cortés (2013). An example of a JSON file is showed in our web page (Marquez-Chamorro & Resinas, 2017).

### 5.2. *Runtime predictions*

The software for runtime predictions can be divided into two types of components, namely: predictor and connector. A predictor encapsulates a prediction model and provides a generic API to make predictions of running instances following the approach detailed in Section 4.1. Therefore, once a prediction model has been obtained in the training phase it has to be deployed into a predictor in order to use it at runtime. To deploy a prediction model into a predictor, there are two aspects that must be configured: the encoding that uses, which must match the one used in the prediction model, and the type of prediction model (e.g. whether it is a decision tree, a decision rule set or a regression function). In the current implementation, only window-based encodings and rule set prediction models are available, but new encodings and types of models can be easily added.

The other component of the runtime prediction software are connectors. A connector is used to integrate the predictions provided by predictors into a specific process-aware information system. This enables process participants to make decisions based on the predictions obtained. For instance, if the process-aware information system is a business process management system like Camunda (Camunda, 2016), the connector can be implemented as a Camunda task list plug-in that, together with the information of the process instance at hand, it shows the value of the predictions for all the indicators defined for that process so that the process participants can act accordingly. Fig. 4 shows our Camunda plug-in for process indicator predictions using an example of running process instance.

## 6. Experimentation

This section presents the experimentation results obtained by the evolutionary approach. The aim of this experimentation consists on providing an analysis of the effectiveness of our system with respect to other machine learning approaches. In particular, our evaluation focuses on the following research questions:

- RQ1. Does the proposed approach provide reliable results in terms of prediction with respect to other similar machine learning approaches?
- RQ2. Does the proposed encoding provide enough information for a good performance of a learning classifier?
- RQ3. Which is the best event window size for the encoding of the datasets considering the quality of the results?
- RQ4. Which is the most adequate configuration for the evolutionary algorithm according to the quality of the results?
- RQ5. Which imbalanced class technique is the most convenient for the datasets considering to the quality of the results?

The first question is related to the quality of the results provided by the proposed evolutionary algorithm. The second one takes into account if the proposed encoding provides reliable results. The third one is related to the selection of the size of the window with the aim of testing the quality of the results when computed with different window sizes. Finally, the fourth and fifth questions refer to the aim of finding the best configuration of the EA and the best imbalanced class technique respectively, for the presented datasets in terms of performance results. In the following, we describe the experiments carried out to answer these research questions.

The rest of the section is organized as follows. The characteristics of the datasets used in the experimentation and the preprocessing stage of these datasets are presented in Sections 6.1 and 6.2, respectively. Then, we define the evaluation measures used for testing the effectiveness of methods in Section 6.3. In Section 6.4, a discussion of the obtained results is provided. Finally, the solution set of resulting rules is described in Section 6.5.

### 6.1. *Datasets*

Event logs used in our experimentation were extracted from the BPI Challenge 2013 (Verbeek, 2013) and from the IT Department of Health Services of Andalusia, Spain.

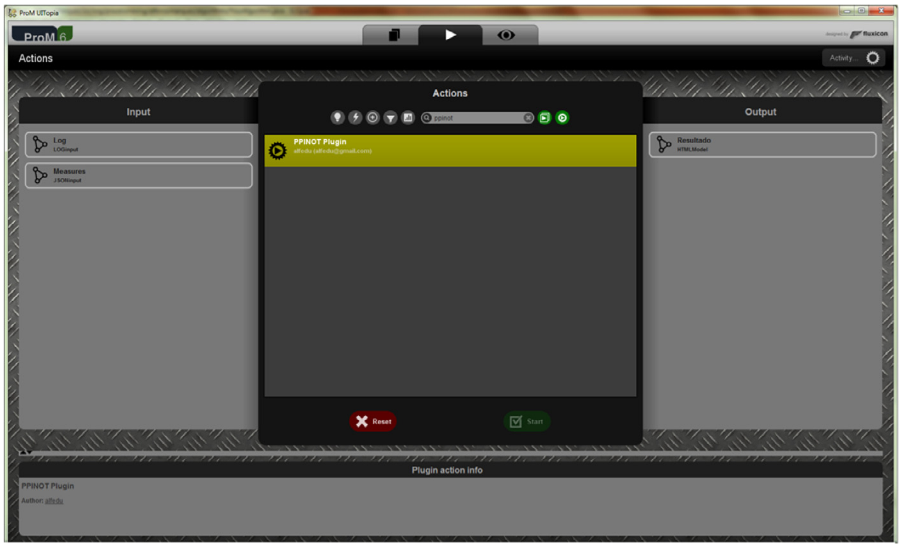
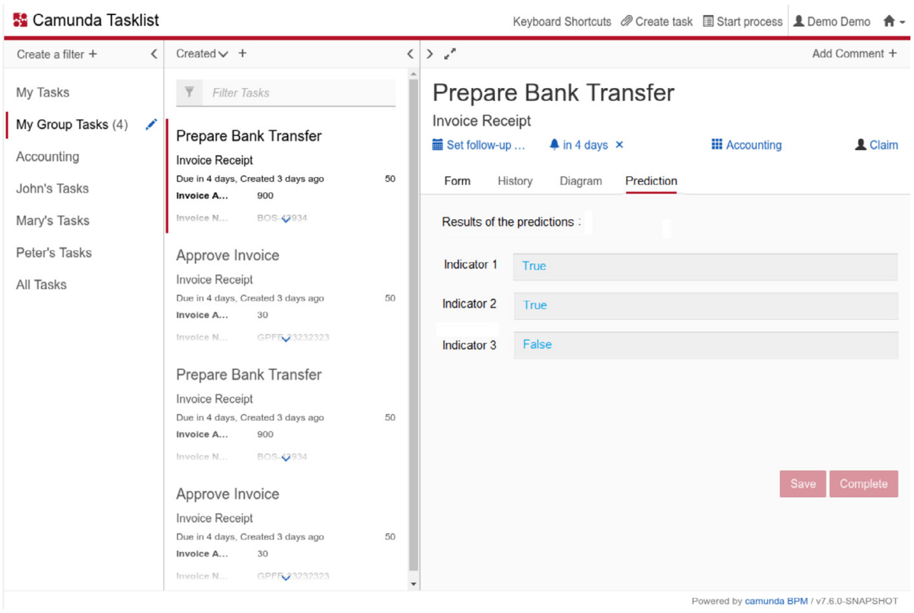


**Fig. 3.** ProM plugin for the calculation of process indicators.



My Tasks
My Group Tasks (4)
Accounting
John's Tasks
Mary's Tasks
Peter's Tasks
All Tasks

Prepare Bank Transfer
Invoice Receipt
Due in 4 days, Created 3 days ago 50
Invoice A... 900
Invoice N... BOS-43934

Approve Invoice
Invoice Receipt
Due in 4 days, Created 3 days ago 50
Invoice A... 30
Invoice N... GPFR-23232323

Prepare Bank Transfer
Invoice Receipt
Due in 4 days, Created 3 days ago 50
Invoice A... 900
Invoice N... BOS-43934

Approve Invoice
Invoice Receipt
Due in 4 days, Created 3 days ago 50
Invoice A... 30
Invoice N... GPFR-23232323

Prepare Bank Transfer
Invoice Receipt
Set follow-up ... in 4 days Accounting Claim

Results of the predictions :

| | |
|---|---|
| Indicator 1 | True |
| Indicator 2 | True |
| Indicator 3 | False |

Powered by camunda BPM / v7.6.0-SNAPSHOT

**Fig. 4.** Example of our Camunda plugin for process indicator predictions.

### 6.1.1. *BPI Challenge 2013 dataset*

Volvo IT incident management log (Repository, 2013) contains all the information of the management of incidents registered at the Volvo IT department. A solution should be established for each incident in order to restore the service with minimum disruption to the business. After providing a solution to the problem and verifying that the service is restored, the incident is closed. Event attributes refers to the support teams involved in the different incidents, the date and time of each event, the impact of the incident and the status of the event among others. The cited dataset consists of 67,543 training instances which belong to 7554 different cases (process instances). The total number of attributes for each training instance is 12. The attributes used for each event are SR Number, Change Date+Time, Status, Sub Status, Involved ST Function Div, Involved Org line 3, Involved ST, SR Latest Impact, Product, Country, Owner Country and Owner First Name. A training instance consists of a list of these attributes for each of the three events and the class attribute (1 for positive cases and 0 for negative cases). An aggregated attribute, named $ChangeDate + Time$, has been calculated as the number of seconds from the beginning of the process until the execution of this event. In this context, this challenge poses three different risk situations to be predicted.

In the first one, a *Push to Front* (PTF) mechanism reveals that most of the incidents need to be resolved by the first line support teams (mainly service desks). *Push to front* works if the 1st line support team can resolve the incident without interference of a 2nd or 3rd line support team. Therefore, we have classified those cases which have finished using only the 1st line support team and those which have been resolved with the interference of the 2nd or 3rd line support team. Our method must predict which cases have to be more likely to be resolved using the 2nd or 3rd support teams.

A second problem to be predicted, is the *PingPong* behavior in the company: the support teams start to send incidents to each other again and again, *e.g.* the 1st line support team sends an incident to the 2*nd* team (Ping) and viceversa (Pong), and consequently the total life time of the incident is increased. In this case, we predict the *PingPong* situations, *i.e*. 1st line and also 2*nd* or 3*rd* and again 1st support line teams take action in the solution of an incident.

A third situation presented in this challenge is the abuse of *waituser* substatus. There exists a PPI that measures the resolution time of the incident. With the use of this substatus, the timing clock stops. Unless someone is really waiting for an end-user, this substatus should not be used by the action owners according to the company guidelines. In this case, the abuse of this substatus in inappropriate situations justify the prediction.

Therefore, we have defined three different process indicators to be predicted in the experimentation: the system will alert from the risk of a PTF fail situation, a Ping Pong situation or a *Waituser* substatus situation.

According to the definitions provided in Section 2, the three indicators stated in this section are classified as instance-level indicators, since they provide a value for each executed trace. Next-event prediction is considered for these indicators because they represent a value for the following event.

#### 6.1.2. IT Department of Andalusian Health Service dataset

This dataset reflects the incident management log of the IT Department of Andalusian Health Service. In this context, a service level agreement (SLA) is established considering certain key performance indicators (KPIs). This SLA determines the penalties derived from the under-fulfilling of a threshold for each of the KPIs. Thus, predictive monitoring is necessary to warn the possibility of violation of the SLA. In this case, two KPIs are considered: K1 determines if an incident is solved in time and K6 determines if an incident has been reopened because it was not correctly solved. Furthermore, three priority levels are also considered for each incident, i.e. normal (P3), high (P2) and very high (P1). The higher the level of priority, the higher the penalties in case of violation of SLA. For K1, priority levels are related with the resolution time of the incidents. P1 means that the resolution time must be lower or equal than 2 h. P2 establish that the resolution time must be lower or equal than 4 h. Finally, P3 indicates that the resolution time should be lower or equal than 17 h. The different process instances are classified according to their priority levels. Since we have considered three different priorities for each KPI, we have obtained 6 different datasets, named K1_P1, K1_P2, K1_P3, K6_P1, K6_P2 and K6_P3.

This dataset consists of 1,055,128 training instances or events. For the K1 indicator and according to the type of priority, we have found 11,254 instances for P1, 18.310 instances for P2 and 1,015,133 instances for P3. On the other hand, for the K6 indicator, we have considered 5425 instances for P1, 10,785 instances for P2 and 852,539 instances for P3. The total number of attributes for each event in the training instance is 14. The attributes used for each event are Autor, Nodo, Resolutor, Descorg, Tipologia, Asunto, Asignatario, Prioridad, GrupoAutor, Estado, Centro, Origen, TypeOrg, Recurso and MotivoCierre. A training instance consists of a list of these attributes and the class attribute. In this case, the two cited process indicators (K1 and K6) for the prediction of SLA violations have been defined for the experimentation.

According to the definitions provided in Section 2, the two indicators stated in this section are classified as instance-level indicators, since they provide a value for each executed trace. For these indicators, end-of-instance prediction is used, since we are predicting the value of the indicator at the end of the process.

### 6.2. Preprocessing of datasets

This section provides a description of the preprocessing stage for the different datasets presented in the previous section.

Considering the event window size for the encoding, we have performed several analysis, described in the Window size experimentation section, and determine that best results were obtained for $N = 3$ for the BPI Challenge and $N = 2$ for the IT Department datasets.

For the PTF dataset, the class is defined according to the attribute *Involved ST* of the event that follows the events included in the window. If the value of this attribute is 2nd or 3rd and it is different for the previous event, then we determine a fail in the PTF mechanism (positive class) and set the class attribute to 1. Otherwise, we set the class attribute to 0.

For the Ping Pong dataset, the class is also defined according to the attribute *Involved ST* of the following event for the Ping and also for the Pong situation. If the value of this attribute is 1st, 2nd or 3rd and it is different for the previous event in the Ping situation, and this value changes to the original support team in the Pong situation, we define a positive class and set the class attribute to 1. The class attribute is set to 0 in the rest of the cases.

Finally, for the *wait user* substatus dataset, the class is defined according to the attribute *Substatus* of the following event. If the value of this attribute is set to *wait user*, then we set the class attribute to 1 (positive class).

For the K1 dataset, the class attribute determine if a process instance was solved in time (negative class) or not (positive class). For the K6 dataset, the class attribute determines if the incident was reopened (positive class) or not (negative class). In this case, the definition and calculation of indicators both have been performed using PPINOT tool (del Río-Ortega, Resinas, Cabanillas, & Ruiz-Cortés, 2013) and the ProM plugin described Section 5.1.

Analyzing all the datasets, we have noticed that the positive and negative classes are notably imbalanced. This is a common problem which affect to machine learning due to having disproportionate number of positive and negative class instances, that lead to provide a misleading classification accuracy. There are many approaches to deal with this problem. Standing out, the re-sampling that can be classified in two methodologies: over-sampling and under-sampling. Over-sampling consists in adding copies of instances to the under represented class. By contrast, under-sampling consists in deleting instances from the over-represented class. We have tested several imbalanced class techniques for the two resulting datasets, and finally, we have obtained the best results using the following methods. For the BPI challenge dataset, we have performed an under-sampling of data using 1:1 and 2:1 ratios. The optimization of this parameter also implies a lower computational cost for the algorithm. On the other hand, for the IT Department dataset, the Borderline SMOTE method (Han, Wang, & Mao, 2005) was applied as over-sampling method. This algorithm selects these samples using the nearest neighbor approach. Experimentations for the justification of these decisions, are provided in Section 6.4.

### 6.3. Effectiveness evaluation

The effectiveness of our proposal is assessed using several measures. We have computed precision, recall, specificity and F_measure defined as shown in Eqs. (3), (4), (5) and (6) respectively. These measures are widely used for testing machine learning approaches in literature.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$Specificity = \frac{TN}{TN + FP} \tag{5}$$

$$F\_measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \qquad (6)$$

*TP* represent true positives, *FP* are false positives, *TN* indicate true negatives and *FN* are false negatives. Precision is the ratio of correctly predicted instances divided by the total of predicted instances. In our case, this measure represents the ratio of instances that have been correctly predicted, over the total of predicted positive instances. Recall or true positive rate indicates what percentage of correctly predicted instances for the positive class have been correctly identified. In our method, this measure shows the ratio of instances that have been correctly predicted over the total of instances. Specificity or true negative rate represents the ratio of correctly predicted instances for the negative class have been correctly identified. In this case, specificity indicates the ratio of correctly predicted instances which have been correctly predicted over the total of instances. F_measure represents the weighted average of the precision and recall.

In the case of imbalanced data, a reliable indicator is area under the curve (AUC) ROC (He & Ma, 2009). The AUC provides a single measure of a classifier performance for evaluating which model is better on average. The ROC curve graphic allows the visualization of the trade-off between the true positives rate and the false positive rate. The true positive rate (TPR) formula is TP/TP+FN which is equivalent to the Recall and represents the benefits of the algorithm. On the other hand, false positive rate (FPR) is equal to FP/FP+TN which is equivalent to $(1 - specificity)$ and represents the cost of the algorithm. The diagonal line represents a random classifier. Points above the diagonal represent good classification results (better than random). Points below the diagonal represent poor classification results (worse than random).

## 6.4. Results

In order to answer the five research questions, we have performed the following experiments described in this section. All the experimentation was performed using the KEEL tool (Alcala-Fdez et al., 2011) and the following classifiers: artificial neural network (ANN)(Broomhead & Lowe, 1988), random trees (RT)(Gray & Fan, 2008), support vector machine (SVM)(Cortes & Vapnik, 1995) and C4.5 decision tree (DT)(Quinlan, 1993). Settings used for the experimentation are as follows, for the decision tree, a confidence factor used for pruning of 0.25 and a minimum number of object per leaf of 2. For the random trees, we have established 0 randomly chosen attributes at each node and 1 as minimum total weight of instances in a leaf. For the multilayer perceptron, we have set the learning rate to 0.3, the momentum to 0.2, the training time to 500 epochs, the hidden layers is equal to $(attribs + classes)/2$ and the validation threshold was set to 20. Finally, for the support vector machine, we have selected the polynomial kernel with an exponent value of 1.0, the complexity parameter $c$ is set to 0.1, the tolerance parameter is set to 0.001 and $\varepsilon$ is set to $1.0^{-12}$. Available datasets can be downloaded from our web (Marquez-Chamorro & Resinas, 2017).

The obtained results can be seen in Tables 2–7 for a 10-fold cross-validation partitioning scheme. The experiments were also performed with the aim of validating our representation and confirms that the new encoding provides enough information for a good performance of a learning classifier. Moreover, we can also notice that the EA, based on the algorithms described in Corcoran and Sen (1994) and Aguilar-Ruiz, Riquelme, and Toro (2003), achieved good accuracy results for all datasets in contrast to the other algorithms as shown in Tables 2–7.

**Table 3**
Average results obtained for the different classification algorithms for PTF behavior dataset with balance ratio 2:1 (a) and 1:1 (b), for Ping Pong behavior dataset with balance ratio 1:1 (c) and for *Wait user* substatus dataset with balance ratio 1:1 (d), respectively

| (a) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
|---|---|---|---|---|
| DT | $0.821_{\pm0.05}$ | $0.585_{\pm0.05}$ | $\mathbf{0.932}_{\pm0.04}$ | 0.657 |
| RT | $0.807_{\pm0.08}$ | $0.670_{\pm0.07}$ | $0.840_{\pm0.03}$ | 0.732 |
| ANN | $0.492_{\pm0.03}$ | $0.599_{\pm0.02}$ | $0.673_{\pm0.06}$ | 0.540 |
| SVM | $\mathbf{0.958}_{\pm0.09}$ | $0.523_{\pm0.08}$ | $0.890_{\pm0.07}$ | 0.676 |
| EA | $0.874_{\pm0.01}$ | $\mathbf{0.758}_{\pm0.01}$ | $0.930_{\pm0.02}$ | **0.811** |
| (b) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
| DT | $\mathbf{0.950}_{\pm0.03}$ | $0.398_{\pm0.06}$ | $\mathbf{0.982}_{\pm0.07}$ | 0.560 |
| RT | $0.884_{\pm0.07}$ | $0.722_{\pm0.08}$ | $0.782_{\pm0.05}$ | 0.795 |
| ANN | $0.610_{\pm0.01}$ | $0.722_{\pm0.01}$ | $0.577_{\pm0.02}$ | 0.661 |
| SVM | $0.930_{\pm0.11}$ | $0.507_{\pm0.08}$ | $0.897_{\pm0.10}$ | 0.656 |
| EA | $0.916_{\pm0.2}$ | $\mathbf{0.868}_{\pm0.01}$ | $0.899_{\pm0.03}$ | **0.891** |
| (c) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
| DT | $\mathbf{0.963}_{\pm0.07}$ | $0.580_{\pm0.05}$ | $0.965_{\pm0.09}$ | 0.723 |
| RT | $0.830_{\pm0.03}$ | $0.724_{\pm0.06}$ | $0.913_{\pm0.06}$ | 0.784 |
| ANN | $0.495_{\pm0.10}$ | $0.415_{\pm0.12}$ | $0.760_{\pm0.08}$ | 0.451 |
| SVM | $0.580_{\pm0.04}$ | $0.500_{\pm0.04}$ | $\mathbf{0.987}_{\pm0.05}$ | 0.537 |
| EA | $0.895_{\pm0.01}$ | $\mathbf{0.753}_{\pm0.03}$ | $0.930_{\pm0.04}$ | **0.817** |
| (d) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
| DT | $0.892_{\pm0.08}$ | $\mathbf{0.870}_{\pm0.03}$ | $0.960_{\pm0.07}$ | 0.880 |
| RT | $0.837_{\pm0.02}$ | $0.725_{\pm0.05}$ | $0.853_{\pm0.06}$ | 0.776 |
| ANN | $0.315_{\pm0.12}$ | $0.274_{\pm0.10}$ | $0.778_{\pm0.09}$ | 0.293 |
| SVM | $0.425_{\pm0.05}$ | $0.603_{\pm0.03}$ | $0.730_{\pm0.06}$ | 0.498 |
| EA | $\mathbf{0.960}_{\pm0.02}$ | $0.772_{\pm0.04}$ | $\mathbf{0.981}_{\pm0.03}$ | **0.855** |

**Table 4**
Average results obtained for the different classification algorithms for K1 dataset and different priorities: P1 (a), P2 (b) and P3 (c), respectively.

| (a) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
|---|---|---|---|---|
| DT | $\mathbf{0.933}_{\pm0.02}$ | $0.727_{\pm0.03}$ | $0.974_{\pm0.03}$ | 0.817 |
| RT | $0.827_{\pm0.04}$ | $0.622_{\pm0.03}$ | $0.972_{\pm0.03}$ | 0.709 |
| SVM | $0.838_{\pm0.08}$ | $0.737_{\pm0.09}$ | $\mathbf{0.978}_{\pm0.07}$ | 0.784 |
| ANN | $0.742_{\pm0.09}$ | $0.570_{\pm0.07}$ | $0.933_{\pm0.07}$ | 0.644 |
| EA | $0.875_{\pm0.06}$ | $\mathbf{0.830}_{\pm0.05}$ | $\mathbf{0.978}_{\pm0.05}$ | **0.851** |
| (b) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
| DT | $0.833_{\pm0.12}$ | $0.791_{\pm0.04}$ | $0.984_{\pm0.03}$ | 0.811 |
| RT | $0.712_{\pm0.08}$ | $0.595_{\pm0.09}$ | $0.976_{\pm0.03}$ | 0.648 |
| SVM | $0.848_{\pm0.07}$ | $0.778_{\pm0.09}$ | $0.986_{\pm0.05}$ | 0.811 |
| ANN | $0.853_{\pm0.11}$ | $0.513_{\pm0.05}$ | $0.994_{\pm0.06}$ | 0.471 |
| EA | $\mathbf{0.885}_{\pm0.07}$ | $\mathbf{0.801}_{\pm0.08}$ | $\mathbf{0.993}_{\pm0.07}$ | **0.840** |
| (c) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
| DT | $0.700_{\pm0.04}$ | $0.513_{\pm0.03}$ | $\mathbf{0.890}_{\pm0.03}$ | 0.592 |
| RT | $0.661_{\pm0.05}$ | $0.503_{\pm0.05}$ | $0.871_{\pm0.05}$ | 0.571 |
| SVM | $0.689_{\pm0.07}$ | $0.596_{\pm0.09}$ | $0.865_{\pm0.08}$ | 0.639 |
| ANN | $0.584_{\pm0.9}$ | $0.338_{\pm0.08}$ | $0.764_{\pm0.09}$ | 0.428 |
| EA | $\mathbf{0.898}_{\pm0.09}$ | $\mathbf{0.749}_{\pm0.12}$ | $\mathbf{0.890}_{\pm0.10}$ | **0.816** |

**Table 5**
Average results obtained for the different classification algorithms for K6 dataset and different priorities: P1 (a), P2 (b) and P3 (c), respectively.

| (a) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
|---|---|---|---|---|
| DT | $0.886_{\pm0.08}$ | $0.855_{\pm0.03}$ | $\mathbf{0.977}_{\pm0.06}$ | 0.855 |
| RT | $0.793_{\pm0.07}$ | $0.541_{\pm0.04}$ | $\mathbf{0.977}_{\pm0.08}$ | 0.643 |
| SVM | $0.950_{\pm0.10}$ | $0.810_{\pm0.05}$ | $0.965_{\pm0.06}$ | 0.874 |
| ANN | $0.511_{\pm0.03}$ | $0.541_{\pm0.06}$ | $0.950_{\pm0.07}$ | 0.525 |
| EA | $\mathbf{0.960}_{\pm0.03}$ | $\mathbf{0.875}_{\pm0.05}$ | $0.954_{\pm0.12}$ | **0.915** |
| (b) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
| DT | $0.865_{\pm0.05}$ | $\mathbf{0.847}_{\pm0.03}$ | $\mathbf{0.994}_{\pm0.06}$ | 0.855 |
| RT | $0.846_{\pm0.06}$ | $0.588_{\pm0.05}$ | $0.993_{\pm0.05}$ | 0.693 |
| SVM | $0.844_{\pm0.05}$ | $0.786_{\pm0.06}$ | $0.943_{\pm0.04}$ | 0.814 |
| ANN | $0.754_{\pm0.04}$ | $0.557_{\pm0.07}$ | $0.818_{\pm0.05}$ | 0.640 |
| EA | $\mathbf{0.886}_{\pm0.06}$ | $0.829_{\pm0.06}$ | $0.955_{\pm0.09}$ | **0.856** |
| (c) | $Precision_{\mu\pm\sigma}$ | $Recall_{\mu\pm\sigma}$ | $Specificity_{\mu\pm\sigma}$ | $F\_measure$ |
| DT | $0.871_{\pm0.08}$ | $0.472_{\pm0.07}$ | $\mathbf{0.995}_{\pm0.02}$ | 0.612 |
| RT | $0.776_{\pm0.04}$ | $0.506_{\pm0.04}$ | $0.941_{\pm0.03}$ | 0.612 |
| SVM | $0.872_{\pm0.06}$ | $0.649_{\pm0.06}$ | $0.952_{\pm0.05}$ | 0.744 |
| ANN | $0.735_{\pm0.07}$ | $0.497_{\pm0.06}$ | $\mathbf{0.995}_{\pm0.04}$ | 0.593 |
| EA | $\mathbf{0.898}_{\pm0.08}$ | $\mathbf{0.849}_{\pm0.09}$ | $0.934_{\pm0.10}$ | **0.872** |

**Table 6**
Average accuracy results for the prediction of next event using BPI Challenge 2013 dataset.

| Method | Precision | Recall | Specificity | *F_measure* |
|---|---|---|---|---|
| History | 0.569 | 0.308 | 0.961 | 0.399 |
| 2-gram | 0.621 | 0.346 | 0.965 | 0.444 |
| 3-gram | 0.633 | 0.368 | 0.966 | 0.465 |
| 4-gram | 0.635 | 0.377 | **0.967** | 0.473 |
| 5-gram | 0.631 | 0.378 | 0.966 | 0.472 |
| 6-gram | 0.624 | 0.375 | 0.966 | 0.468 |
| EA | **0.972** | **0.692** | 0.791 | **0.808** |

**Table 7**
Average accuracy results obtained for different sizes of the event window.

| Size | *Precision*$_\mu$*PTF* | *Recall*$_\mu$*PTF* | *Precision*$_\mu$*K1_P1* | *Recall*$_\mu$ *K1_P1* |
|---|---|---|---|---|
| 2 | $0.907_{\pm 0.05}$ | $0.856_{\pm 0.02}$ | $\mathbf{0.875}_{\pm 0.02}$ | $\mathbf{0.830}_{\pm 0.05}$ |
| 3 | $\mathbf{0.916}_{\pm 0.02}$ | $\mathbf{0.868}_{\pm 0.06}$ | $0.795_{\pm 0.09}$ | $0.792_{\pm 0.08}$ |
| 4 | $0.897_{\pm 0.01}$ | $0.847_{\pm 0.01}$ | $0.774_{\pm 0.10}$ | $0.750_{\pm 0.08}$ |
| 5 | $0.885_{\pm 0.05}$ | $0.835_{\pm 0.03}$ | $0.760_{\pm 0.08}$ | $0.743_{\pm 0.09}$ |

#### 6.4.1. *Results for BPI Challenge 2013 dataset*

In the following, we present the results for the prediction of the three different process indicators cited in Datasets section (PTF fail, Ping Pong and *wait user*).

**Push-to-front (PTF) behavior.** Table 3(a) shows the average results and standard deviations achieved for the classification algorithms for the PTF dataset with balance ratio 2:1. We can appreciate that best values of recall and F_measure are achieved using the EA. Although DT obtains similar values of specificity, this method only predicts a 58.5% of the positive cases (recall) in contrast to the 75.8% achieved by the EA. SVM obtains a good precision rate of 95.8%, however the recall rate is low (52.3%), in contrast to our method that achieves a good 87.4% and 75.8% of precision and recall rates respectively. The F_measure of our method obtains 8 points above the second best rate.

Table 3(b) presents the results for the PTF dataset with balance ratio 1:1. DT obtains good results of precision and specificity, however, the recall rate is widely improved by our algorithm (86.8% against 39.8%). We can also appreciate that our F_measure exceeds substantially (more than 33 points approximately) the equivalent rate for DT. Although the SVM approach achieves similar results for precision and specificity, the true positive rate is low in contrast to the recall rate achieved by the EA approach (50.7% against 86.8%). Low values of standard deviation of our method, show us that our results are not significantly spread. Recall rate and F_measure achieved by EA overcomes the equivalent results of the rest of approaches.

Generally, we can appreciate that recall rates of the EA widely improved the rates achieved by the other proposals. One of the purposes of the prediction in this area is the prevention of undesirable situations (*i.e.* fails in the PTF mechanism in our case). In that sense, a low value of recall means that there is a high number of fails in the PTF mechanism that the algorithm will not be able to detect. Therefore, the high recall obtained by the EA means that our algorithm cover the highest number of situations without PTF.

Fig. 5(a) and (b) shows the ROC curve for the results of the four best classifiers for both PTF datasets (2:1 and 1:1). As we can see, the higher value is widely achieved by the EA in both cases.

**Ping Pong behavior.** Table 3(c) shows the results for the Ping Pong dataset with balance ratio 1:1. F_measure and recall rates achieved by our algorithm are the highest values in comparison to the other approaches. Although precision rate of DT overcomes the one achieved by our EA, our recall rate exceeds in is almost 20 points to DT recall rate. That means that our algorithm identifies 75.3% (recall) of Ping Pong cases and from that percentage, a 89% (precision) of the cases are correctly identified. Specificity achieved by the EA is among the three best rates. Fig. 5 (c) presents the AUC graphic for the results of the four best classifiers. The outcomes show that the value of AUC of the EA improves the value of the other classifiers.

***Wait user* substatus problem.** Table 3(d) presents the results for the prediction of the *Wait user* substatus dataset with balance ratio 1:1. Best results of precision and specificity are achieved with our algorithm. In this case, recall rate of EA is overcome by DT method (0.870 against 0.772), although the rate of correctly predicted *Wait user* situations (precision) of the EA, exceeds in 7 points to the DT precision. F_measure of our method is the second best rate (0.85) of the rest of predictors. Fig. 5 (d) shows the ROC curve for the results of the four best classifiers. As we can see, the higher value of AUC is achieved by the EA.

#### 6.4.2. *Results for IT Department of Andalusian Health Service dataset*

Two indicators have been predicted for this dataset. K1 predicts if a process instance is going to be solved in time. K6 determines the possibility of an incident will be reopened after being closed. The prediction of these indicators attempt to prevent the non-compliance of the established SLAs.

**Prediction of K1 indicator.** Table 4 shows the average results and standard deviations achieved for the classification algorithms for the K1 dataset and the different priorities. Best values of F_measure, specificity and recall are achieved using the EA for the three priorities. Precision also obtains best values for P2 and P3 priorities. We can highlighted that, for P3 priority in Table 4 (c), recall rate exceeds in 15 points to the second best rate classifier (SVM) and the precision obtained by the EA achieves a 89.8% which is considered a high value with respect to the 70.0% obtained by the DT. Therefore, in this case, we have achieved the highest differences with respect to the rest of the classifiers. This could be due to the fact that this is the largest dataset and EAs are good classifiers finding suboptimal solutions in high dimensionality search spaces. Fig. 6 shows for the three different priorities that the higher value of AUC is achieved by the EA.

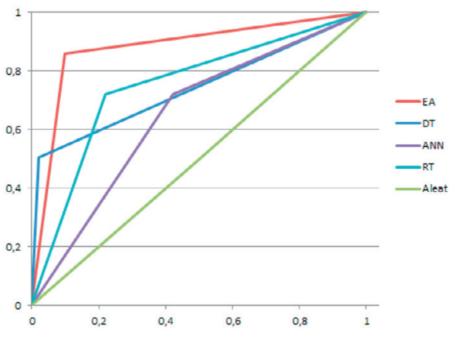
**Prediction of K6 indicator.** Table 5 shows the average results and standard deviations achieved for the classification algorithms for the K6 dataset and the different priorities. For all the priorities, precision and F_measure values of the EA, widely improves the rest of the classifiers rates. The EA also obtains best values of recall for P1 and P3 priorities and for P2 priority, we can observe that DT obtain best recall rate, but is closely followed by our EA. Fig. 7 shows that AUC value of the EA, improves the value of the other classifiers in this case.

According to the obtained results in this and previous sections, we can provide a positive response to RQ1 research question. We have obtained reliable results for the EA which improve, in most cases, the outcomes of the other machine learning techniques. Specifically, EA outperforms the performance rates of the other approaches in 8/10 experiments for recall, 8/10 for specificity, and 6/10 for precision and 10/10 for F_measure. We can conclude that our EA obtains good recall rates in average with respect to the other classifiers because EA is adequate for working with huge dimensional search spaces.

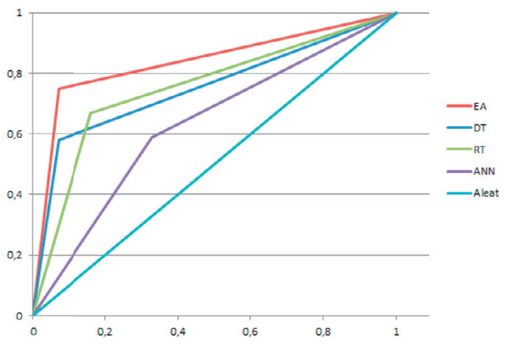
We can also conclude that this encoding provide enough information for a good performance of a learning classifier. This is evidenced by the good results achieved by all the machine learning classifiers. In this case, we can also provide a positive answer to RQ2.

#### 6.4.3. *Comparative analysis*

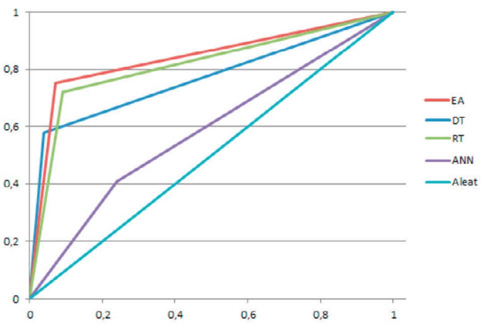
We have performed a comparative analysis with one of the recent works of the literature (Breuker, Matzner, Delfmann, & Becker,



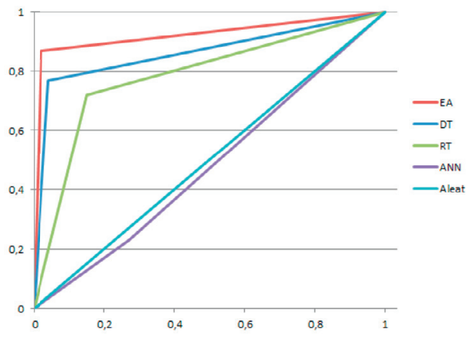
(a) PTF behavior dataset (2:1).



(b) PTF behavior dataset (1:1).



(c) Ping Pong behavior dataset (1:1).



(d) Wait user substatus dataset (1:1).

**Fig. 5.** ROC curves for the different classification algorithms for PTF behavior, Ping Pong behavior and Wait user substatus datasets with balance ratios.

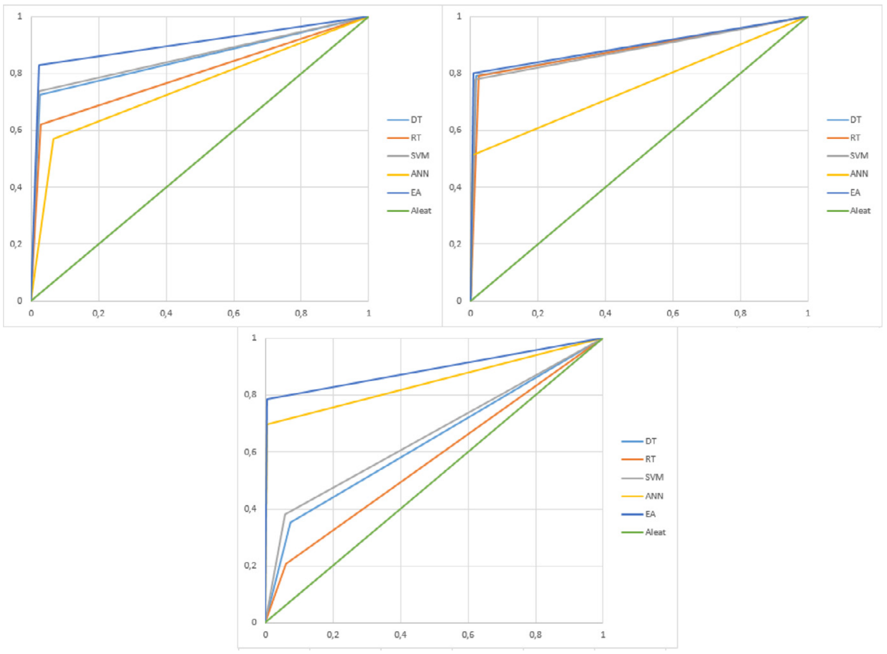


**Fig. 6.** ROC curves for the different classification algorithms for K1 dataset. First graph on the left correspond to P1 priority. Second graph on the right belongs to P2 priority and graph at the bottom corresponds to P3 priority.

2016) which also utilizes BPI Challenge 2013 dataset. In this case, the predicted outcome is the following event of the running process instance. We have defined the events with the properties Status and Substatus and their possible values, i.e. Queued, Accepted and Completed for Status and In progress, Awaiting assignment, Assigned, Wait - User, Closed, Wait - Implementation, Resolved, In call, Wait - Customer and Wait - Vendor for Substatus. A 3-event window was selected for the encoding. Results are shown in Table 6. The method named *History* uses as input the entire sequence of events, whereas *n-gram* methods take into account a sliding window of size n of events, as seen in Breuker, Matzner, Delfmann, and Becker (2016). Our method, named EA, widely exceeds all the results for precision, recall and F_measure in more than 30 percentage points.

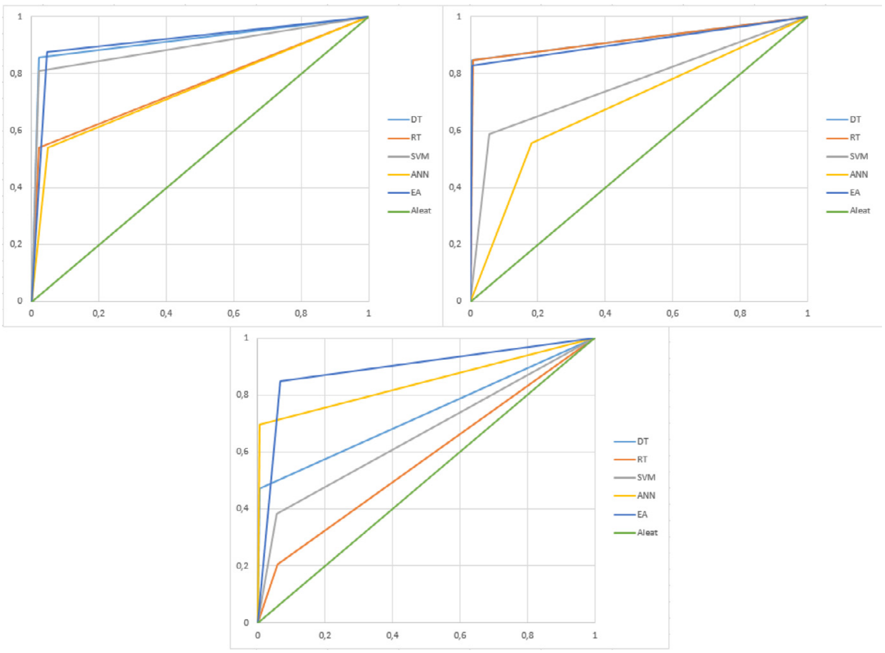


**Fig. 7.** ROC curves for the different classification algorithms for K6 dataset. First graph on the left correspond to P1 priority. Second graph on the right belongs to P2 priority and graph at the bottom corresponds to P3 priority.

### 6.4.4. *Window size experimentation*

In order to answer the third research question, we have performed an experiment using different window sizes for the encoding. Specifically, we have varied the window size from 2 to 5 events. Different results for precision and recall rates are shown in Table 7. All the experimentations were performed with the PTF dataset with balance ratio 1:1 and using K1 dataset with priority P1 in representation of each one of the two datasets previously described. As can be seen in Table 7, the window of three events provides the best recall and precision performances for the PTF dataset. On the other hand, K1 dataset achieves its best recall and precision rates for a 2-window size. Furthermore, a window size of 2 and 3 implies a lower computational cost. Based on the results of the table, we adopt these sizes of window for our encoding in order to answer to RQ3.

### 6.4.5. *Evolutionary algorithm settings*

One of the challenges using an EA is to find the best configuration of the evolutionary parameters. In this experimentation, we have tested several configurations as result of modifying the probability values of evolutionary operators (crossover and mutation) and the number of generations of each execution in order to answer RQ4. According to the Table 8, best precision results were obtained using a probability of 0.8 for crossover. Fixing the crossover probability, we have obtained best results for a mutation probability of 0.5. Although we appreciate that some configurations for crossover, achieve better recall rates (*e.g.* probability of 1.0), maximum precision rate is reached using these settings. Concerning the number of generations, we can notice a slight improvement using 300 generations, however the execution time was moderately increased with this modification, and we finally chose 100 as the number of generations. For those reasons, and considering RQ4, we conclude that most adequate configuration for the evolutionary algorithm according to the quality of the results is probability of 0.5 and 0.8 for the mutation and crossover operators respectively, and 100 as number of generations.

**Table 8**
Average accuracy results obtained for different configurations of the EA using K1_P1 dataset.

| Cross. prob. | $Precision_{\mu}$ | $Recall_{\mu}$ |
|---|---|---|
| 0.5 | $0.869_{\pm 0.05}$ | $0.830_{\pm 0.02}$ |
| 0.6 | $0.786_{\pm 0.02}$ | $0.821_{\pm 0.06}$ |
| 0.7 | $0.764_{\pm 0.01}$ | $0.827_{\pm 0.01}$ |
| 0.8 | $\mathbf{0.875}_{\pm 0.02}$ | $0.830_{\pm 0.05}$ |
| 0.9 | $0.816_{\pm 0.01}$ | $0.840_{\pm 0.01}$ |
| 1.0 | $0.816_{\pm 0.05}$ | $\mathbf{0.842}_{\pm 0.03}$ |
| Mutat. prob. | $Precision_{\mu}$ | $Recall_{\mu}$ |
| 0.5 | $\mathbf{0.875}_{\pm 0.02}$ | $\mathbf{0.830}_{\pm 0.05}$ |
| 0.6 | $0.750_{\pm 0.02}$ | $0.790_{\pm 0.06}$ |
| 0.7 | $0.764_{\pm 0.03}$ | $0.817_{\pm 0.01}$ |
| 0.8 | $0.764_{\pm 0.04}$ | $0.805_{\pm 0.03}$ |
| 0.9 | $0.845_{\pm 0.01}$ | $0.825_{\pm 0.01}$ |
| 1.0 | $0.742_{\pm 0.07}$ | $0.801_{\pm 0.03}$ |
| Generations | $Precision_{\mu}$ | $Recall_{\mu}$ |
| 100 | $0.875_{\pm 0.02}$ | $0.830_{\pm 0.05}$ |
| 200 | $0.816_{\pm 0.03}$ | $0.818_{\pm 0.03}$ |
| 300 | $\mathbf{0.897}_{\pm 0.08}$ | $\mathbf{0.847}_{\pm 0.07}$ |
| 400 | $0.875_{\pm 0.03}$ | $0.825_{\pm 0.03}$ |
| 500 | $0.875_{\pm 0.05}$ | $0.825_{\pm 0.04}$ |

### 6.4.6. *Imbalanced class techniques comparison*

Different unbalanced classes techniques have been applied to our datasets in order to obtain the best prediction results and to answer RQ5. Imbalanced techniques can be divided into four groups: over-sampling methods, under-sampling methods, cost-sensitive methods and algorithmic modifications for class imbalance. We have analyzed the different over-sampling and under-sampling techniques in the literature using the K1_P1 dataset. Table 9 shows the performance rates for the different imbalanced methods. Best results were achieved using Borderline SMOTE (Han, Wang, & Mao, 2005), an over-sampling method that generates positive examples from other instances of the original training dataset selecting k nearest neighbors. Thus, we can conclude that the most convenient imbalanced class technique for our datasets considering the performance results is Borderline SMOTE.

**Table 9**
Average accuracy results obtained for different imbalanced class techniques.

| Over-sampling | *Precision*$_\mu$ | *Recall*$_\mu$ |
|---|---|---|
| ADASYN (He, Bai, Garcia, & Li, 2008) | $0.588_{\pm 0.05}$ | $0.338_{\pm 0.02}$ |
| SPIDER (Stefanowski & Wilk, 2008) | $0.600_{\pm 0.02}$ | $0.264_{\pm 0.06}$ |
| Bline-SMOTE (Han, Wang, & Mao, 2005) | $\mathbf{0.897}_{\pm 0.01}$ | $\mathbf{0.847}_{\pm 0.01}$ |
| ROS (Kubat & Matwin, 1997) | $0.777_{\pm 0.05}$ | $0.529_{\pm 0.03}$ |
| SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) | $0.636_{\pm 0.01}$ | $0.617_{\pm 0.01}$ |
| **Under-sampling** | *Precision*$_\mu$ | *Recall*$_\mu$ |
| CNN (Hart, 1968) | $0.619_{\pm 0.05}$ | $0.477_{\pm 0.02}$ |
| CPM (Yoon & Kwek, 2005) | $0.575_{\pm 0.02}$ | $0.507_{\pm 0.06}$ |
| NCL (Laurikkala, 2001) | $0.714_{\pm 0.01}$ | $0.225_{\pm 0.01}$ |
| OSS (Kubat & Matwin, 1997) | $0.549_{\pm 0.05}$ | $0.529_{\pm 0.03}$ |
| RUS (Batista, Prati, & Monard, 2004) | $0.507_{\pm 0.01}$ | $0.461_{\pm 0.01}$ |

## 6.5. Resulting rules

The predictive system has generated a total of 20 rules on average for each fold. Two examples of rules generated by the algorithm, one that predicts a fail in the *Push to front* mechanism, and one that reflects the possibility of violation of the SLA defined by K1 indicator, are showed below:

$$\begin{aligned}&if\ Status\_2 \in [accepted, waiting]\ and\ InvolvedSTFunctionDiv\_1\\&= [A24]\ and\ InvolvedSTFunctionDiv\_2 = [A24]\\&and\ InvolvedOrgLine3_3 = OrgLineA2\ and\ SRLatestImpact\_2 \in\\&[medium, high]\ and\ Product\_1 \in [PR582, PR583]\ and\ Country\_3\\&= France\ then\ yes\ Accuracy = 0.893 \qquad (7)\end{aligned}$$

If we inspect this rule, a fail in the *Push to front* mechanism is more likely to occur when the *status* of the second event is *accepted* or *waiting* and the *InvolvedSTFunctionDiv* of the first and second events is *A*24 and *InvolvedOrgLine*3 of the third event is equal to *OrgLineA*2 and *SRLatestImpact* of the second event is *medium* or *high* and the *Product* of the first event is *PR*582 or *PR*583 and the *Country* of third event is *France*.

$$\begin{aligned}&if\ Resolutor\ \in [CEGES\ GESTION\ SERVICIO\ \ldots\ SAS\ OFIMATICA]\\&and\ Tipologia\ \in [Incidencia, Incidencia.Sistema]\ and\\&Asignatario\ = H.\ U.\ REINA\ SOFIA\ and\ MotivoCierre\\&= Recurso.Reparado\ and\ TypeOrg2 \in [RE, CL]\ and\ Nodo2\\&\in [SSCC, CADIZ]\ and\ Prioridad2 = P3\ and\ Origen2\\&\in [TELEFONO, INTRANET]\ then\ false\ Accuracy = 0.822 \qquad (8)\end{aligned}$$

This second rule reflects that, a violation of the SLA is more likely to happen when, for the first event, *Resolutor* is *CEGES GESTION* or *SAS OFIMATICA* and *Tipologia* is *Incidencia* or *Incidencia. Sistema* and *Asignatario* is equal to *H. U. REINA SOFIA I* and *MotivoCierre* is equal to *Recurso.Reparado* and for the second event, *TypeOrg*2 is *RE* or *CL* and *Nodo*2 is *SSCC* or *CADIZ* and *Prioridad*2 is equal to *P*3 and *Origen*2 is *TELEFONO* or *INTRANET*. We can notice that rules generated by our proposal are characterized by a high reliability, accuracy values of 0.893 and 0.822, and also high interpretability, which would allow users to easily inspect the results for further validation.

Some conclusions can be extracted from these generated rules. We can analyze the relevant attributes for the prediction according to their number of appearances in the rules. We can also analyze the rules from different perspectives of the characteristics of the processes to extract further conclusions. For instance, the first rule determines a mostly use of attributes referred to the organizational perspective, *e.g. InvolvedSTFunctionDiv* and *InvolvedOrgLine*3 which define the name of the team that executes the activity and the organizational level, respectively. The resource/organizational perspective was used for process mining in Park et al. (2015). Second rule determine the use of context attributes, such as *Tipologia*, which determines the typology of the IT incident, or *Nodo* related to the location of the company. Context information was also involved in several process mining works, such as do Espírito-Santo-Carvalho, Santoro, and Revoredo (2015).

## 7. Conclusions and future work

In this paper an evolutionary decision rule-based system for the prediction of business process indicators is described. The encoding of this approach is based on the attributes of the events extracted from the event logs. The decision rules determine a prediction of a specified process indicator. Our system can predicts instance-level indicators using both next-event and end-of-instance predictions. Unlike the prediction models obtained by other machine learning techniques, these generated decision rules can be easily interpreted by users to extract further insight of the business processes and improve the running process. An interpretable model is important in order to identify relevant features or find out the reasons of performance problems in the process. For instance, rules described in previous section define the situations to avoid the PTF problem. Results presented in this work using two real-life event logs, show the validity of the proposal. Obtained results in terms of recall, precision and specificity, are perceived superior in comparison to the other machine learning approaches. The development of a ProM plugin for the definition and calculation of process indicators also represents a significative contribution of this work. As far as we are concerned, any other software is able to define business process indicators and provide the calculation of their values for the different process instances of an event log. The different datasets used in the experimentation of this work are collected in our web page.

Future steps will involve identifying alternative technologies for the prediction such as deep learning which is becoming in a thriving research topic. In this sense, recurrent neural networks can be used to predict process indicators, as seen in Evermanna, Rehseb, and Fettkeb (2016). We trust that a comparative analysis between our evolutionary method and a deep-learning-based one may provide helpful insights about the applicability and usefulness of deep learning on predictive analytics. Furthermore, an extended ProM plugin with the prediction method will also be developed. Finally, an analysis of root causes of performance issues, based on Li, Joe-Qin, and Yuan (2016b), will be performed in order to find new predicted indicators.

## Acknowledgements

This work has received funding from the European Commission (FEDER), the Spanish and the Andalusian R+D+I programmes, BELI and COPAS, [grant numbers TIN2015-70560-R, P12TIC-1867] and Juan de la Cierva program [JCF 2015].

## References

Aguilar-Ruiz, J. S., Riquelme, J. C., & Toro, M. (2003). Evolutionary learning of hierarchical decision rules. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 33(2)*, 324–331.

Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garcia, S., Sanchez, L., & Herrera, F. (2011). KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing, 17*, 255–287.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations, 6:1*, 20–29.

Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys, 49*(2), 31:1–31:50.

Bratosin, C., Sidorova, N., & van der Aalst, W. M. P. (2010). Discovering process models with genetic algorithms using sampling. In *Proceedings of the 14th international conference on knowledge-based and intelligent information and engineering systems: Part I* (pp. 41–50).

Breuker, D., Matzner, M., Delfmann, P., & Becker, J. (2016). Comprehensible predictive models for business processes. *MIS Quarterly, 40*(4), 1009–1034.

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptative networks. *Complex Systems, 11*, 321–355.

Cabanillas, C., Ciccio, C. D., Mendling, J., & Baumgrass, A. (2014). Predictive task monitoring for business processes. In *Business process management*. In *Lecture Notes in Computer Science: 8659* (pp. 424–432).

Camunda (2016). An open source platform for workflow and business process management. Accessed: 09.03.17. https://camunda.org/.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Conforti, R., de Leoni, M., Rosa, M. L., & van der Aalst, W. M. P. (2013). Supporting risk-informed decisions during business process execution. In *Advanced information systems engineering: 7908* (pp. 116–132).

Conforti, R., de Leoni, M., Rosa, M. L., van der Aalst, W. M. P., & ter Hofstede, A. H. M. (2015). A recommendation system for predicting risks across multiple business process instances. *Decision Support Systems, 69*, 1–19.

Corcoran, A. L., & Sen, S. (1994). Using real-valued genetic algorithms to evolve rule sets for classification. In *1st IEEE conference on evolutionary computation. Orlando, USA* (pp. 120–124).

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning, 20*, 273–297.

van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., & van der Aalst. , W. (2005). The ProM framework: A new era in process mining tool support. In *Applications and theory of petri nets* (pp. 444–454).

do Espírito-Santo-Carvalho, J., Santoro, F. M., & Revoredo, K. (2015). A method to infer the need to update situations in business process adaptation. *Computers in Industry, 71*, 128–143.

Evermanna, J., Rehseb, J. R., & Fettkeb, P. (2016). Predicting process behaviour using deep learning. *Decision Support Systems, In press*.

Fogel, D.B. (1997). The advantages of evolutionary computation,.

Folino, F., Guarascio, M., & Pontieri, L. (2012). Discovering context-aware models for predicting business process performances. In *Proceedings of 20th international conference on cooperative inf. systems (CoopIS12)* (p. 287-304).

Francescomarino, C. D., Dumas, M., Maggi, F. M., & Teinemaa, I. (2015). Clustering-based predictive process monitoring. *CoRR, abs/1506.01428*.

Gray, J. B., & Fan, G. (2008). Classification tree analysis using target. *Computational Statistics and Data Analysis, 52(3)*, 1362–1372.

Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing (ICIC'05). Lecture notes in computer science 3644, Springer-Verlag 2005, Hefei, China* (pp. 878–887).

Hart, P. E. (1968). The condensed nearest neighbour rule. *IEEE Transactions on Information Theory, 14:5*, 515–516.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *International joint conference on neural networks (IJCNN08)* (pp. 1322–1328).

He, H., & Ma, Y. (2009). *Imbalanced learning: foundations, algorithms, and applications*. Wiley.

Kamsu-Foguem, B., Rigal, F., & Mauget, F. (2013). Mining association rules for the quality improvement of the production process. *Expert Systems with Applications, 40*, 1034–1045.

Kang, B., Kim, D., & Kang, S. (2012). Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction. *Expert Systems with Applications, 39*(5), 6061–6068.

Karray, M. H., Chebel-Morello, B., & Zerhouni, N. (2014). PETRA: Process evolution using a trace-based system on a maintenance platform. *Knowledge-Based Systems, 68*(1), 21–39.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *14th international conference on machine learning (ICML97). Tennessee, USA* (pp. 179–186).

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *8th conference on AI in medicine in Europe (AIME01). LNCS 2001, Springer 2001, Cascais, Portugal* (pp. 63–66).

Leitner, P., Hummer, W., & Dustdar, S. (2013). Cost-based optimization of service compositions. *IEEE Transactions on Services Computing, 6*(2). 2013

de Leoni, M., van der Aalst, W. M. P., & Dees, M. (2016). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Sciences, 56*, 235–257.

Li, C., Ge, J., Huang, L., Hu, H., Wu, B., Yang, H., … Luo, B. (2016a). Process mining with token carried data. *Information Sciences, 328*, 558–576.

Li, G., Joe-Qin, S., & Yuan, T. (2016b). Data-driven root cause diagnosis of faults in process industries. *Chemometrics and Intelligent Laboratory Systems, 159*(15), 1–11.

Maggi, F. M., Francescomarino, C. D., Dumas, M., & Ghidini, C. (2014). Predictive monitoring of business processes. *Advanced information systems engineering. Lecture Notes in Computer Science, 8484, 457–472*.

Marquez-Chamorro, A.E. (2016). Prom plugin for the preprocess stage of predictive monitoring. Accessed: 09.03.17. https://svn.win.tue.nl/repos/prom/Packages/AlfonsoMarquez/Trunk/.

Marquez-Chamorro, A. E., Asencio-Cortes, G., Divina, F., & Aguilar-Ruiz, J. S. (2014). Evolutionary decision rules for predicting protein contact maps. *Pattern Analysis and Applications, 17:4*, 725–737.

Marquez-Chamorro, A.E., & Resinas, M. (2017). Website with supplementary material for this work. Accessed: 09.03.17. http://www.isa.us.es/predictivemonitoring/ea.

Medeiros, A. K., Weijters, A. J. M. N., & van der Aalst, W. M. P. (2007). Genetic process mining: An experimental evaluation. *Data Mining and Knowledge Discovery, 14(2)*, 245–304.

Nkambou, R., Fournier-Viger, P., & Mephu-Nguifo, E. (2011). Learning task models in ill-defined domain using an hybrid knowledge discovery framework. *Decision Support Systems, 24(1)*, 176–185.

Park, M., Song, M., Baek, T. H., Son, S., Ha, S. J., & Cho, S. W. (2015). Workload and delay analysis in manufacturing process using process mining. In *Asia Pacific business process management - third Asia Pacific conference, AP-BPM 2015, Busan, South Korea, June 24-26* (pp. 138–151).

Pika, A., van der Aalst, W. M. P., Fidge, C. J., ter Hofstede, A. H. M., & Wynn, M. T. (2013). Predicting deadline transgressions using event logs. In *Business process management workshops: 132* (pp. 211–216).

Pika, A., van der Aalst, W. M. P., Wynn, M. T., Fidge, C. J., & ter Hofstede, A. H. M. (2016). Evaluating and predicting overall process risk using event logs. *Information Sciences, 352-353*, 98–120.

Polato, M., Sperduti, A., Burattin, A., & de Leoni, M. (2014). Data-aware remaining time prediction of business process instances. *International joint conference on neural networks (IJCNN14).*

Potes-Ruiz, P., Kamsu-Foguem, B., & Grabot, B. (2014). Generating knowledge in maintenance from experience feedback. *Knowledge-Based Systems, 68*, 4–20.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kauffman, San Mateo-California.

Repository, BPIC (2013). BPI challenge 2013 repository. Accessed: 09.03.17. https://data.3tu.nl/repository/uuid:500573e6-accc-4b0c-9576-aa5468b10cee.

del Rio-Ortega, A. (2017). PPINOT. tools for the definition and automated analysis of process performance indicators. Accessed: 09.03.17. http://www.isa.us.es/ppinot.

del Río-Ortega, A., Resinas, M., Cabanillas, C., & Ruiz-Cortés, A. (2013). On the definition and design-time analysis of process performance indicators. *Information Systems, 38*(4), 470–490.

del Río-Ortega, A., Resinas, M., Durán, A., Bernárdez, B., Ruiz-Cortés, A., & Toro, M. (2017). Visual PPINOT: A graphical notation for process performance indicators. *Business & Information Systems Engineering, tbd*(tbd), tbd.

Rogge-Solti, A., & Weske, M. (2015). Prediction of business process durations using non-Markovian stochastic petri nets. *Information Systems, 54*, 1–14.

Schnig, S., Cabanillas, C., Jablonski, S., & Mendling, J. (2016). A framework for efficiently mining the organisational perspective of business processes. *Decision Support Systems, 89*, 87–97.

Stefanowski, J., & Wilk, S. (2008). Selective pre-processing of imbalanced data for improving classification performance. In *10th international conference on data warehousing and knowledge discovery (DaWaK08). Turin, Italy* (pp. 283–292).

Tax, N., Verenich, I., Rosa, M. L., & Dumas, M. (2016). Predictive business process monitoring with LSTM neural networks. *CoRR, abs/1612.02130*.

Verbeek, H.M.W. (2013). Third international business process intelligence challenge (BPIC 13). Accessed: 09.03.17. https://www.win.tue.nl/bpi/doku.php?id=2013:challenge/.

Wen, J., Zhong, M., & Wang, Z. (2015). Activity recognition with weighted frequent patterns mining in smart environments. *Expert Systems with Applications, 42*(17-18), 6423–6432.

Yoon, K., & Kwek, S. (2005). An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. In *5th international conference on hybrid intelligent systems (HIS05), Rio de Janeiro (Brazil, 2005)* (pp. 303–308).