Complex Symbolic Sequence Clustering and Multiple Classifiers for Predictive Process Monitoring

Ilya Verenich[1,2], Marlon Dumas[2], Marcello La Rosa[1], Fabrizio Maria Maggi[2], and Chiara Di Francescomarino[3]

[1] Information Systems School, Queensland University of Technology, Australia
{ilya.verenich,m.larosa}qut.edu.au
[2] Institute of Computer Science, University of Tartu, Estonia
{marlon.dumas,f.m.maggi}@ut.ee
[3] FBK-IRST, Trento, Italy
dfmchiara@fbk.eu

Abstract. This paper addresses the following predictive business process monitoring problem: Given the execution trace of an ongoing case, and given a set of traces of historical (completed) cases, predict the most likely outcome of the ongoing case. In this context, a trace refers to a sequence of events with corresponding payloads, where a payload consists of a set of attribute-value pairs. Meanwhile, an outcome refers to a label associated to completed cases, like, for example, a label indicating that a given case completed "on time" (with respect to a given desired duration) or "late", or a label indicating that a given case led to a customer complaint or not. The paper tackles this problem via a two-phased approach. In the first phase, prefixes of historical cases are encoded using complex symbolic sequences and clustered. In the second phase, a classifier is built for each of the clusters. To predict the outcome of an ongoing case at runtime given its (uncompleted) trace, we select the closest cluster(s) to the trace in question and apply the respective classifier(s), taking into account the Euclidean distance of the trace from the center of the clusters. We consider two families of clustering algorithms – hierarchical clustering and k-medoids – and use random forests for classification. The approach was evaluated on four real-life datasets.

Keywords: Process Mining, Predictive Process Monitoring, Complex Symbolic Sequence, Clustering, Ensemble Methods

1 Introduction

Modern business processes are supported by information systems that record data about each individual execution of a process, also referred to as a case. These data can be structured in the form of event logs consisting of traces, each capturing the events produced in the context of one case. Such event logs can be used for various business process analytics tasks [21].

Predictive process monitoring [14] is concerned with exploiting such event logs to predict how running (uncompleted) cases will unfold up to their completion. One particular type of predictive process monitoring is that of estimating the probability that an ongoing case will lead to a certain outcome among a set of possible outcomes. In this context, an outcome could be, for example, the timely completion of the case with respect to a deadline (versus late completion), or the fulfillment of a desired business goal (e.g., a sales process leading to an order, or an issue handling process leading to successful resolution).

Given that event logs consist of sequences of event records (each possibly associated to a corresponding payload), the above predictive monitoring problem can be seen as the one of early sequence classification [27], that is a problem of assigning a "label" (outcome) to a sequence based on: (i) a prefix thereof; and (ii) a set of labeled completed sequences (the "history"). A direct approach to this problem is to extract features from prefixes of historical sequences and use them to train a classifier. This classifier is then used at runtime in order to assign a label to the incomplete trace of an ongoing case.

This paper investigates an alternative cluster-and-classify approach to this predictive monitoring problem. The proposed approach proceeds in two phases. First, prefixes of previous traces are clustered. Secondly, a classifier is built for each cluster to discriminate between different outcomes (e.g., "normal" versus "deviant" cases). At runtime, a prediction is made on a running case by mapping it to one or multiple clusters and applying the corresponding classifier(s).

The paper explores multiple variants of the proposed approach based on two clustering techniques (k-medoids and hierarchical) as well as a single-classifier approach versus a multiple-classifier (ensemble) approach. These variants are experimentally compared using as baseline a plain classication-only approach.

The paper is organized as follows. In Section 2, we provide a brief survey of previous work on the predictive process monitoring. Section 3 presents the proposed method for predictive process monitoring. The validation is discussed in Section 4. Finally, Section 5 draws conclusions and outlines possible future work.

2 Background and Related Work

This section provides a review of existing predictive business process monitoring approaches. Most of these approaches are based on sequence classification methods.

Many previous works deal with the problem of identifying and eliminating process-related risks. For example, Pika et al. [17] make predictions about time-related process risks, by identifying and exploiting indicators observable in event logs that affect the likelihood of violating specified deadlines. Conforti et al. [3] propose a technique to reduce possible process risks by supporting the process participants in making risk-informed decisions. Risks are predicted by traversing decision trees generated from the logs of past process executions. Suriadi et al. [20] present an approach for Root Cause Analysis through classification

algorithms. Decision trees are used to retrieve the causes of overtime faults on a log enriched with information about delays, resources and workload. Metzger et al. [15] present a technique for predicting "late show" events in transportation processes. Specifically, they apply standard statistical techniques to find correlations between "late show" events and external variables related to weather conditions or road traffic. Grigori et al. [8] present an approach and a tool suite for real-time exception analysis, prediction, and prevention.

Another group of works deals with the time perspective. In [25], van Dongen et al. develop an approach for predicting the remaining cycle time of a case by using non-parametric regression with case-related data as predictor variables. In [22,23], van der Aalst et al. present a set of approaches in which annotated transition systems, containing time information extracted from event logs, are used to check time conformance while cases are being executed, predict the remaining processing time of incomplete cases, and recommend appropriate activities to end users working on these cases. Rogge-Solti and Weske [18] use stochastic Petri nets to predict the remaining execution time of a process, taking into account the time passed since the last observed process event. Folino et al. [6] develop a predictive clustering approach, where various context-related execution scenarios are discovered and modeled via distinct state-aware performance predictors. A predictive model is obtained eventually that can make performance forecasts for any new running test case.

Zeng et al. [29] adopt the ARIMA forecasting method to predict performance criteria for event sequences. The approach is applied for aggregated key performance indicators (KPI) rather than single instances. Then classification is applied to separate cases that meet KPIs from those that violate them. Kang et al. [9] propose an approach for predicting abnormal termination of business processes. They apply a fault detection technique based on k-nearest neighbor algorithm to estimate the probability that a fault occurs. Alarms are generated for an early notification of probable abnormal terminations. Lakshmanan et al. [11] develop a technique to estimate the probability of execution of any potential future task in an ongoing process instance using extended Markov chain. Xing et al. [26] mine a set of sequential classification rules as a classifier. An evaluation was conducted based on simple DNA sequence datasets and the method was demonstrated to be very accurate. Greco et al. [7] approach the issue of classifying a process instance using frequent itemset detection.

Closely related to predictive monitoring is deviance mining [5]. While predictive monitoring deals with predicting the impact of actions and decisions of process participants on the outcomes of ongoing process executions, deviance mining deals with the offline analysis of process execution logs in order to detect common abnormal executions and to explain deviance that leads to increased or decreased performance [19]. Together, these two techniques provide evidence-based management of business processes, which enables process participants to receive guidance to achieve required process outcomes and performance [5].

Most of the above mentioned works rely either on the control-flow or on the data perspective for making predictions at runtime, but they do not take

both perspectives into consideration. The two perspectives have been considered together only by Maggi et al. [14], where a framework was proposed to predict whether or not an ongoing case will fulfill a given predicate upon its completion based on: (i) the sequence of activities executed in a given case; and (ii) the values of data attributes after each execution of an activity in a case. This framework has been shown to be relatively accurate, but at the expense of high runtime overhead, since the classifiers used by the model are constructed at runtime. Thus this framework is not applicable in settings with high throughput or when instantaneous response times are required to help users make rapid decisions.

In order to overcome this problem, the framework has been extended in [4] by introducing a clustering pre-processing phase which allows for the pre-computation of the classification models. This allows for a drastic reduction of the prediction time and for facing high throughput loads. Nevertheless, a limitation of this approach is that only the payload of the last executed event is taken into account, while neglecting the evolution of data values throughout the execution traces. In [13], this limitation is addressed by treating execution traces as complex symbolic sequences and processing them as such. Our research extends the approach presented in [4] through the use of a multiple classifier method and the combination of the clustered-based approach with the approach based on complex symbolic sequences introduced in [13].

3 Approach

We propose a two-phased approach for predictive process monitoring. In the first phase, from a log of historical (i.e., completed) cases, we extract prefixes of a fixed length n and encode them using complex symbolic sequences. In particular, we use the index-based encoding presented in [13]. Thus, we obtain feature vectors that can be clustered. In the second phase, we use the sequences contained in each cluster to train a classifier. In this work, we apply random forest. Random forest is a powerful classifier that has been already applied for similar problems [16], [24]. To predict the outcome of an ongoing case at runtime given its (uncompleted) trace, we select the closest cluster(s) to the trace in question (taking into account the Euclidean distance of the trace from the center of the clusters) and apply the respective classifier(s).

3.1 Sequence Encoding

In the proposed approach, an execution trace is treated as a complex symbolic sequence, i.e., a sequence of events each carrying a data payload consisting of event attributes. In order to apply the clustering approach, trace prefixes need to be encoded in terms of a feature vector. Complex sequences can be encoded as feature vectors in several ways. In this paper, we use an encoding based on indexes as proposed by Leontjeva et al. in [13], which has been shown to yield a relatively high accuracy. In particular, index-based encoding specifies for each position in a trace, the event occurring in that position and the value of each data attribute in that position.

3.2 Clustering

Clustering is a type of unsupervised learning technique in which a structure has to be devised on top of unlabeled data. The main idea behind clustering is organizing a dataset into groups (clusters), so that elements within a cluster are more similar to each other than elements belonging to different clusters. Many clustering algorithms have been proposed in the literature, as well as a number of possible dimensions for their classification. In this work, we apply two clustering approaches – hierarchical clustering and k-medoids.

Hierarchical Agglomerative Clustering. Hierarchical agglomerative clustering (HAC) belongs to a family of clustering algorithms that measure the distance between clusters based on the distances between pairs of elements (e.g., maximum or minimum distance between two elements of the clusters) [28]. In particular, hierarchical agglomerative clustering groups data elements into a tree of clusters (dendrogram), building it in a bottom-up fashion. It starts by placing each individual data element in its own cluster and then merges these atomic clusters into larger clusters. This process continues until all data elements are gathered in a single cluster or certain termination conditions are satisfied.

In hierarchical clustering, clusters are defined as branches of a cluster tree. The constant height branch cut, a commonly used method to identify branches of a cluster tree, is not ideal for cluster identification in complicated dendrograms. In this work, we use *adaptive* hierarchical clustering as defined by Langfeldera et al. [12]. They describe a new dynamic branch cutting approach for detecting clusters in a cluster tree based on their shape.

k-medoids Clustering. In k-medoids clustering (KM), clusters are represented by the so-called medoid element and, hence, the distance between clusters is based on the distance between their medoid elements. The k-medoids algorithm selects first k data elements, which become the medoids elements. The algorithm then assigns each of the remaining data elements to the clusters, based on the distance between the element and the cluster medoid. Then, a new medoid for each cluster is computed. This process iterates until convergence has been reached. The k-medoids algorithm is closely related to the k-means algorithm.

3.3 Classification

Random Forest. The random forest is an ensemble classifier that consists of a large number of randomly trained decision trees. To classify a case, each tree outputs its prediction, or "vote", and the final decision is determined by majority voting [1].

The performance of the random forest is linked to the level of correlation between any two trees in the forest. The lower the correlation is, the higher the overall performance of the entire forest is [2].

Multiple Classifier Method. For classifying an ongoing case at runtime, generally, we select the closest cluster, run the classifier that has been trained for that cluster and output the probability of a case belonging to class "normal" or class "deviant". The closest cluster is chosen based on the smallest distance of the point representing the case from each center. However, it is often the case that the point is almost equally distant from two or more centers. To accommodate this situation, we take the output of each classifier with a weight that is inversely proportional to the distance from the point to the center of the cluster. For example, if we have two clusters, and a point is equally distant from the center of each cluster, then, we take the output of each classifier with weight 0.5.

4 Evaluation

4.1 Setup and Datasets

We conducted the experiments on four real-life datasets. Table 1 summarizes the characteristics of the logs (number of normal and deviant cases, average case length, total number of events, and total number of event classes).

Dataset	Normal cases	Deviant cases	Total cases	Median trace length	Num of events	Event classes
$BPIC_{\varphi_1}$	743	172	915	44	120,036	623
$BPIC_{\varphi 2}$	232	683	915	44	120,036	623
Hospital	448	363	811	18	14,825	26
Insurance	788	277	1065	12	16,869	9

Table 1: Case study datasets.

The first dataset (BPI) is taken from the BPI 2011 challenge and contains events related to treatment and diagnosis steps for patients diagnosed with cancer in a Dutch Academic Hospital. Specifically, each case refers to the treatment of a particular patient. The event log contains domain specific attributes that are both case attributes and event attributes. For example, *Age*, *Diagnosis*, *Diagnosis code*, and *Treatment code* are case attributes, whereas *Activity code*, *Number of executions*, *Specialism code*, and *Group* are event attributes. From this log, we define a deviance as a violation of a compliance rule. In particular, we use the following linear temporal logic rules [14]:

- $\varphi_1 = \mathbf{G}(\text{“}CEA-tumor\ marker\ using\ meia\text{”} \rightarrow \mathbf{F}(\text{“}squamous\ cell\ carcinoma\ using\ eia\text{”}))$
- $\varphi_2 = \mathbf{F}(\text{“}historical\ examination-big\ resectiep\text{”})$

where $\mathbf{F}\varphi$ indicates that φ is true sometimes in the future and $\mathbf{G}\varphi$ means that φ is true always in the future. The rule φ_1 states, hence, that every time "CEA-tumor marker using meia" occurs, then "squamous cell carcinoma using eia"

has to occur eventually, while rule φ_2 states that "historical examination - big resectiep" has to occur eventually at least once.

The second log we used (Hospital) refers to the treatment of patients with chest pain in an emergency department of an Australian hospital. To classify the cases in this log, we used a temporal deviance criterion. In particular, we labeled as quick those cases that complete within 180 minutes and as slow those cases that need more than 180 minutes to complete. We considered the slow cases as deviant. With this definition, the dataset is nearly balanced w.r.t. the class labels – 448 normal cases and 363 deviant cases (Table 1).

The third log (Insurance) is taken from a large Australian insurance company and records an extract of the instances of a commercial insurance claim handling process. In this log, we also used a temporal deviance criterion for classification, marking cases quick if they complete within 30 days, and slow, i.e., deviant otherwise.

It should be noted that cases in the datasets have very different lengths and the values in Table 1 only indicate the median length. Moreover, normal and deviant cases typically have different lengths. Fig. 1 shows the distribution of lengths of normal and deviant cases in each dataset.

4.2 Evaluation Metrics

Most classifiers are capable of outputting not only class labels, but also class probabilities, i.e., probabilities that a data sample belongs to a particular class. In this case, predictive ranking ability can be measured with the *receiver operating characteristic* (ROC) curve. A ROC curve represents *ranked* accuracy. The horizontal axis is the proportion of false positives (FPR), that is negative cases that were incorrectly identified as positives, and the vertical one is the corresponding proportion of true positives (TPR), that is actual positive cases that were correctly identified [10]. For a well-performing classifier this curve would be as closer to the top left corner as possible, thus minimizing FPR, and maximizing TPR. For a random guessing this curve would be diagonal.

The area under a ROC curve (AUC) represents the probability that the binary classifier will score a randomly drawn positive sample higher than a randomly drawn negative sample [10]. A value of AUC equal to 1 indicates a perfect ranking, where any positive sample is ranked higher than any negative sample. A value of AUC equal to 0.5 indicates the worst possible classifier that is not better than random guessing. Finally, a value of AUC equal to 0 indicates a reserved perfect classifier, where all positive samples get the lowest ranks.

4.3 Results

We evaluated 5 different variants obtained by applying the only random forest classification (No clustering) or by combining clustering techniques (KM and HAC) with classification techniques (single and multiple classifiers).

We split each log into training set (80%) and test set (20%). To classify an ongoing case of length n, we select the set of pre-built (offline) clusters containing

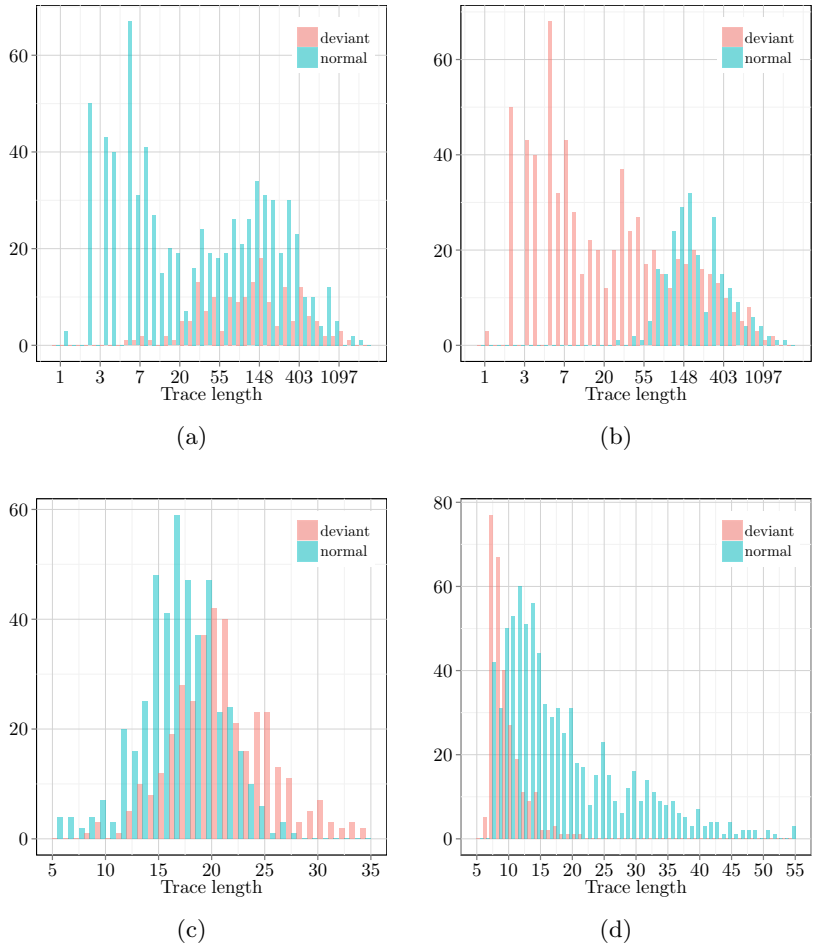

Fig. 1: Distribution of case lengths for the $BPIC_{\varphi_1}$ (a), $BPIC_{\varphi_2}$ (b), *hospital* (c) and *insurance* (d) datasets.

prefixes of length n.[1] Then, we determine cluster that is the closest to the ongoing case. Through the associated classifier, we estimate the probability for the case to end up normally. Considering that in the logs we have full information about the completion of cases also in the test set, we can compare predicted and actual labels and calculate AUC. Fig. 2 shows the average AUC for various prefix lengths, i.e., n varying from 2 to 20. Additionally, Table 2 reports the average AUC values across various prefix lengths.

[1] except for the "No clustering" approach.

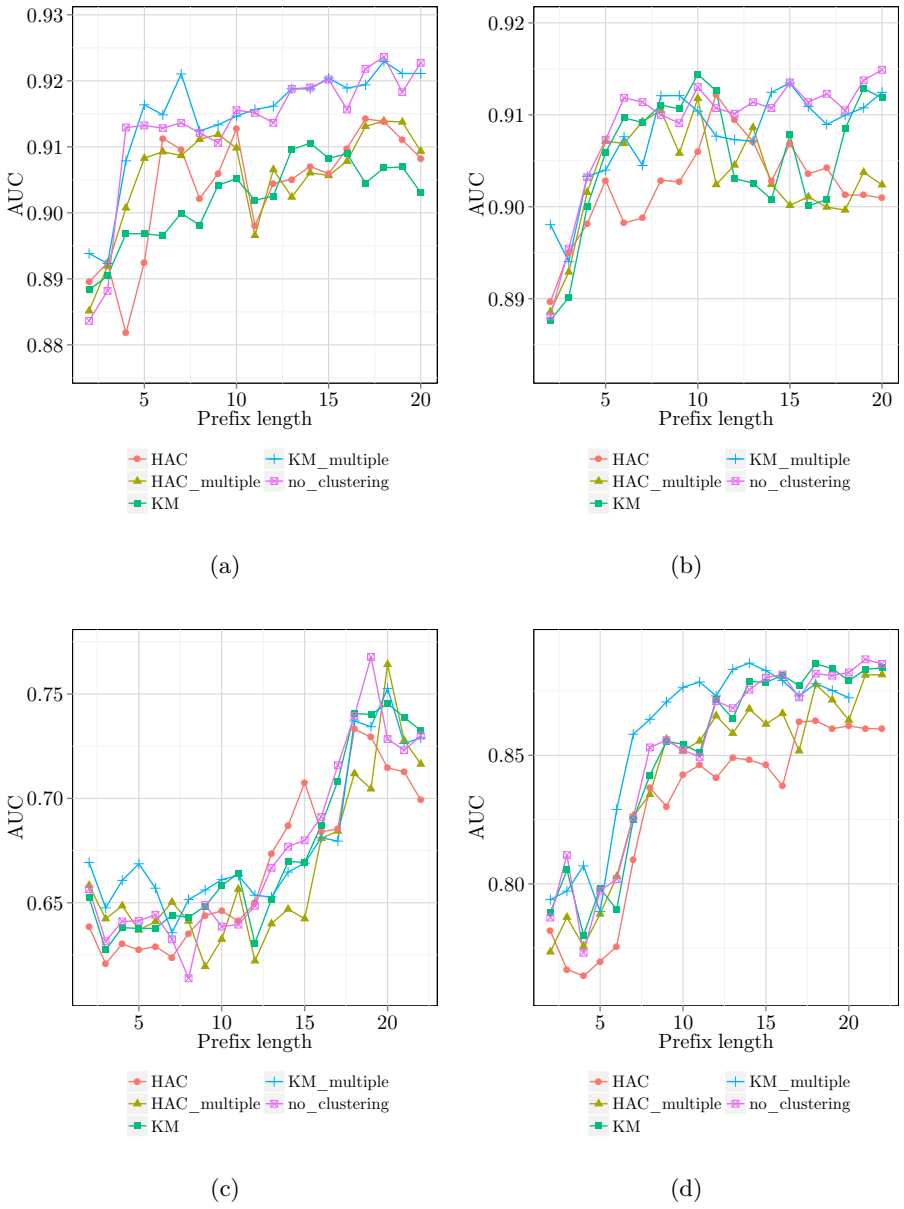

Fig. 2: Mean AUC values using prefixes of different lengths for the $BPIC_{\varphi_1}$ (a), $BPIC_{\varphi_2}$ (b), *hospital* (c) and *insurance* (d) datasets.

Looking at the plots and at the table, we can notice that the multiple-classifiers k-medoids approach outperforms by a small margin other methods,

Method	Prefix sizes 2 to 5				Prefix sizes 2 to 20			
	$BPIC_{\varphi 1}$	$BPIC_{\varphi 2}$	*Hospital*	*Insurance*	$BPIC_{\varphi 1}$	$BPIC_{\varphi 2}$	*Hospital*	*Insurance*
No clustering	0.899	0.898	0.643	0.792	0.913	**0.908**	0.674	0.847
KM	0.893	0.896	0.639	0.793	0.902	0.905	0.675	0.847
KM, multiple cl.	**0.903**	**0.900**	**0.662**	**0.797**	**0.915**	**0.908**	**0.679**	**0.856**
HAC	0.889	0.896	0.629	0.771	0.904	0.902	0.667	0.826
HAC, multiple cl.	0.897	0.898	0.647	0.781	0.906	0.903	0.665	0.839

Table 2: Mean AUC values across various prefix sizes. Best method for each dataset is highlighted.

including random forest without clustering which corresponds to the approach implemented in [13]. Interestingly, for smaller prefix sizes ($2 < n < 10$) its advantage is more evident, but with the increase of n it becomes much less visible.

On the other hand, almost all methods were able to achieve AUC > 0.9 on the *BPIC* datasets with prefix sizes $n \geq 5$, which is about the 10% of the median case length. For the *Insurance* dataset, we achieved AUC > 0.8 for $n \geq 7$, which is about half of the median case length. The result for the *Hospital* dataset was lower (average AUC never exceeded 0.68), probably due to the fact that this dataset includes mostly *case* attributes that do not change over a case, rather than *event* attributes.

For real-time prediction, it is also important to be able to output the results as fast as possible. Thus, we also measured the time needed to classify test cases. The experiments were conducted using `GNU R` version 3.2.0 on a computer with a 2.4 GHz Intel Core i5 quad core processor and 8 Gb of RAM. We found that the average time needed to classify a test case using random forest without clustering is around 0.2 milliseconds for the maximum prefix size $n = 20$. For clustering with multiple classifiers, the prediction time ranges from 20 to 50 milliseconds per case. Therefore, we can conclude that the proposed approach allows for low run-time overhead predictions. Additionally, the time needed to cluster training cases and build offline classifiers for each cluster never exceeds 60 seconds for $n = 20$.

5 Conclusion

The paper has presented a cluster-and-classify approach to address the problem of predicting the outcome of individual cases of a business process. We have explored the design space by considering two clustering techniques (HAC and KM) and by considering the case where an ongoing case is matched against one single classifier (associated to one cluster) or against multiple classifiers (associated to multiple clusters). The evaluation on real-life datasets has shown that the multiple-classifiers k-medoids variant outperforms others (including a pure classification-based approach) in the context of early prediction, i.e., predictions based on short prefixes of a case.

The presented work has several limitations. First, the observations are based on a rather reduced number of datasets, representing two application domains

(hospital healthcare and insurance). A broader evaluation is warranted to back the initial observations. Second, the advantages of the cluster-and-classify approach relative to a pure classification approach, while consistent, are relatively minor. Designing further optimizations of the proposed methods to achieve higher accuracy is a direction for future work. Finally, deeper comparison with alternative methods such as methods based on generative models, e.g., Hidden Markov Models (HMMs) or Conditional Random Fields, could be considered. A systematic comparative evaluation covering this latter technique, the proposed cluster-and-classify method, and other alternative methods is another avenue for future work.

References

1. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.
2. Leo Breiman. Random forests. *Machine Learning*, 45:532, 2001.
3. Raffaele Conforti, Massimiliano de Leoni, Marcello La Rosa, Wil MP van der Aalst, and Arthur HM ter Hofstede. A recommendation system for predicting risks across multiple business process instances. *Decision Support Systems*, 69:1–19, 2015.
4. C. Di Francescomarino, M. Dumas, F. M. Maggi, and I. Teinemaa. Clustering-Based Predictive Process Monitoring. *ArXiv e-prints*, June 2015.
5. Marlon Dumas and Fabrizio Maria Maggi. Enabling process innovation via deviance mining and predictive monitoring. In *BPM-Driving Innovation in a Digital World*, pages 145–154. Springer, 2015.
6. Francesco Folino, Massimo Guarascio, and Luigi Pontieri. Discovering context-aware models for predicting business process performances. In *On the Move to Meaningful Internet Systems: OTM 2012*, pages 287–304. Springer, 2012.
7. Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, and Domenico Sacca. Mining unconnected patterns in workflows. *Information Systems*, 32(5):685–712, 2007.
8. Daniela Grigori, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Improving business process quality through exception understanding, prediction, and prevention. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 159–168, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
9. Bokyoung Kang, Dongsoo Kim, and Suk-Ho Kang. Real-time business process monitoring method for prediction of abnormal termination using knni-based lof prediction. *Expert Systems with Applications*, 39(5):6061–6068, 2012.
10. I. Kononenko and M. Kukar. *Machine Learning and Data Mining*. Elsevier Science, 2007.
11. Geetika T Lakshmanan, Davood Shamsi, Yurdaer N Doganata, Merve Unuvar, and Rania Khalaf. A markov prediction model for data-driven semi-structured business processes. *Knowledge and Information Systems*, 42(1):97–126, 2015.
12. Peter Langfeldera, Bin Zhangb, and Steve Horvatha. Dynamic tree cut: in-depth description, tests and applications. *November*, 22:2007, 2007.
13. Anna Leontjeva, Raffaele Conforti, Chiara Di Francescomarino, Marlon Dumas, and Fabrizio Maria Maggi. Complex symbolic sequence encodings for predictive monitoring of business processes. In *Business Process Management - BPM*. Springer, 2015. to appear.

14. Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. Predictive monitoring of business processes. In *Advanced Information Systems Engineering*, pages 457–472. Springer, 2014.
15. Andreas Metzger, Rod Franklin, and Yagil Engel. Predictive monitoring of heterogeneous service-oriented business networks: The transport and logistics case. In *SRII Global Conference (SRII), 2012 Annual*, pages 313–322. IEEE, 2012.
16. Hoang Nguyen, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Suriadi Suriadi. Mining business process deviance: a quest for accuracy. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, pages 436–445. Springer, 2014.
17. Anastasiia Pika, Wil MP van der Aalst, Colin J Fidge, Arthur HM ter Hofstede, and Moe T Wynn. Predicting deadline transgressions using event logs. In *Business Process Management Workshops*, pages 211–216. Springer, 2013.
18. Andreas Rogge-Solti and Mathias Weske. Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In *Service-Oriented Computing*, pages 389–403. Springer, 2013.
19. Mukhammad Andri Setiawan and Shazia Sadiq. A methodology for improving business process performance through positive deviance. *International Journal of Information System Modeling and Design (IJISMD)*, 4(2):1–22, 2013.
20. Suriadi Suriadi, Chun Ouyang, Wil MP van der Aalst, and Arthur HM ter Hofstede. Root cause analysis with enriched process logs. In *Business Process Management Workshops*, pages 174–186. Springer, 2013.
21. Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer Science & Business Media, 2011.
22. Wil MP van der Aalst, Maja Pesic, and Minseok Song. Beyond process mining: from the past to present and future. In *Advanced Information Systems Engineering*, pages 38–52. Springer, 2010.
23. Wil MP Van der Aalst, M Helen Schonenberg, and Minseok Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011.
24. Sjoerd Van Der Spoel, Maurice Van Keulen, and Chintan Amrit. Process prediction in noisy data sets: a case study in a dutch hospital. In *Data-Driven Process Discovery and Analysis*, pages 60–83. Springer, 2013.
25. Boudewijn F van Dongen, RA Crooy, and Wil MP van der Aalst. Cycle time prediction: when will this case finally be finished? In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 319–336. Springer, 2008.
26. Zhengzheng Xing, Jian Pei, Guozhu Dong, and S Yu Philip. Mining sequence classifiers for early prediction. In *SDM*, pages 644–655. SIAM, 2008.
27. Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40–48, 2010.
28. R. Xu and D. Wunsch. *Clustering*. IEEE Press Series on Computational Intelligence. Wiley, 2008.
29. Liangzhao Zeng, Christoph Lingenfelder, Hui Lei, and Henry Chang. Event-driven quality of service prediction. In *Service-Oriented Computing–ICSOC 2008*, pages 147–161. Springer, 2008.