Process Mining to Forecast the Future of Running Cases

Annalisa Appice[1], Sonja Pravilovic[1,2], and Donato Malerba[1]

[1]Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro
via Orabona, 4 - 70126 Bari - Italy
[2]Montenegro Business School, Mediterranean University Vaka Djurovica
b.b. Podgorica - Montenegro
{annalisa.appice, sonja.pravilovic, donato.malerba}@uniba.it

Abstract. Processes are everywhere in our daily lives. More and more information about executions of processes are recorded in event logs by several information systems. Process mining techniques are used to analyze historic information hidden in event logs and to provide surprising insights for managers, system developers, auditors, and end users. While existing process mining techniques mainly analyze full process instances (cases), this paper extends the analysis to running cases, which have not yet completed. For running cases, process mining can be used to notify future events. This forecasting ability can provide insights for check conformance and support decision making. This paper details a process mining approach, which uses predictive clustering to equip an execution scenario with a prediction model. This model accounts for recent events of running cases to predict the properties of future events. Several tests with benchmark logs investigate the viability of the proposed approach.

1 Introduction

Contemporary systems, ranging from high-tech and medical devices to enterprise information systems and e-business infrastructures, record massive amounts of events by making processes visible. Each event has a name and additional mandatory properties which include the timestamp (i.e. exact date and time of occurrence), the lifecycle transition state (i.e. whether the event refers to a task having been started, completed) and the resource (i.e. name of the originator having triggered the event). In addition, each event may be characterized by several optional properties, such as cost, location, outcome, which are specific for the process. Process mining techniques [6] can be used to analyze event logs in order to extract, modify and extend process models, as well as to check conformance with respect to defined process models.

Thus far, several process mining techniques have been used in the discovery, conformance and enhancement of a variety of business processes [7]. They are mainly used in an off-line fashion and rarely for operational decision support. Historical full cases (i.e. instances of the process which have already completed) are analyzed, while running cases (i.e. instances of the process which have not

completed yet) are rarely processed on-line. However, a new perspective has emerged recently. van der Aalst et al [8] demonstrate that process mining techniques are not necessarily limited to the past, but can also be used for the present and the future. To make this philosophy concrete, they have already presented a set of approaches, which can be used very well for operational decision support. In particular, they propose to mine predictive process models from historic data and use them to estimate the remaining processing time and the probability of a particular outcome for running cases [9].

Embracing this philosophy, we consider another predictive task and detail a process mining approach to forecast future events of running cases. This forecasting service can be used to check conformance and recommend appropriate actions. In particular, we can check whether the observed event fits the forecast behavior. The moment the case deviates, an appropriate actor can be alerted in real time. Similarly, we can use forecasts to notify recommendations proposing/describing activity elements (e.g. resource, activity name, cost), which will comply the process model.

In this paper, we transform the task of event forecasting for running cases into a predictive clustering task [2], where the target variables are the properties of future events expected in running cases, while the predictors are properties of recent events up to a certain time window. This transformation technique is usually known as "time delay embedding" [5] and frequently used in stream data mining, where it is also called sliding window model [3]. Historical cases, transformed with the sliding window model, can be processed off-line so that a predictive clustering tree (PCT) [2] can be mined for the predictive aim. A PCT is a tree structured model, which generalizes decision tree by predicting many labels of an examples at once. In this study, it allows us to reveal in advance properties of future events based on properties of recent time-delayed event elements. The PCT can be used to forecast on-line event elements of any new running case. We have implemented this approach in the context of the ProM [10] framework, and verified its effectiveness in various case studies.

This paper is organized as follows. Section 2 introduces some notations and briefly revises the predictive clustering technique as well as the sliding window model used. Section 3 describes the event-based forecasting approach proposed. Section 4 demonstrates the usefulness of our approach using several benchmark case studies. Section 5 concludes the paper.

2 Basics

In this section, we introduce some basic concepts needed for the event forecasting. We describe event logs, as well as introduce the ideas behind predictive clustering and sliding window model.

2.1 Event log

The basic assumption is that the log contains information about activities executed for specific cases of a certain process type, as well as their durations.

Table 1. A fragment of the event log *repair* [7]. The symbol (*) identifies the optional properties which are specific for the process.

id	name	timestamp	resource	lifecycle	phone type*	defect type*	defect fixed*	number repairs*
1	Register	1970-01-02T12:23	System	complete	-	-	-	-
1	Analyze Defect	1970-01-02T12:23	Tester3	start	-	-	-	-
1	Analyze Defect	1970-01-02T12:30	Tester3	complete	T2	6	-	-
1	Repair (Complex)	1970-01-02T12:31	SolverC1	start	-	-	-	-
1	Repair (Complex)	1970-01-02T12:49	SolverC1	complete	-	-	-	-
1	Test Repair	1970-01-02T12:49	Tester3	start	-	-	-	-
1	Test Repair	1970-01-02T12:55	Tester3	complete	-	-	-	-
1	Inform User	1970-01-02T13:10	System	complete	-	-	-	-
1	Archive Repair	1970-01-02T13:10	System	complete	-	-	true	0
2	Register	1970-01-01T11:09	System	complete	-	-	-	-
2	Analyze Defect	1970-01-01T11:09	Tester2	start	-	-	-	-
2	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Definition 1 (Event, Property). *Let $\mathcal{E}$ be the event domain for a process $\mathcal{P}$. An event ϵ ($\epsilon \in \mathcal{E}$) is characterized by a set of* mandatory properties*, that is, the event corresponds to an activity, has a timestamp which represents date and time of occurrence, is triggered by a resource, refers to a specific lifecycle transition state which can be started or completed. Additionally, it can be characterized by variable* process-specific optional properties *such as cost, location, outcome and so on. Optional properties may also be away in an event.*

An event log is a set of events. Each event in the log is linked to a particular trace and globally unique. A trace in a log represents a process instance (e.g. a customer order or an insurance claim) also referred to as case. Time is non-decreasing in each case in the log.

Definition 2. *A case $\mathcal{C}$ is a finite sequence of events $e \in \mathcal{E}$ such that each event occurs only once (i.e. for $1 \leq i < j \leq |\mathcal{C}|$: $\epsilon(i) \neq \epsilon(j)$) and time is non-decreasing (i.e. $time(\epsilon(i)) \leq time(\epsilon(j))$. A log $\mathcal{L}$ is a bag of cases.*

A fragment of event log is reported in Table 1. A case containing nine events is shown. Each event has the mandatory properties and several optional properties. We note that the value may also lack in an event for an optional property.

2.2 Predictive clustering trees

The task of mining predictive clustering trees is now formalized. *Given*

- a descriptive space $\mathbf{X} = \{X_1, X_2, \ldots X_m\}$ spanned by m independent (or predictor) variables X_j,
- a target space $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_q\}$ spanned by q dependent (or target) variables Y_j,

– a set $\mathcal{T}$ of training examples, $(\mathbf{x_i}, \mathbf{y_i})$ with $\mathbf{x_i} \in \mathbf{X}$ and $\mathbf{y}_i \in \mathbf{Y}$

Find a tree structure τ which represents:

– A set of hierarchically organized clusters on T such that for each $u \in T$, a sequence of clusters $C_1, C_2, \ldots, C_k$ exist for which $u \in C_{i_r}$ and the containment relation $C_1 \supseteq C_2 \supseteq \ldots \supseteq C_k$ is satisfied. Clusters $C_1, C_2, \ldots, C_k$ are associated to the nodes $t_1, t_2, \ldots, t_k$, respectively, where each $t_i \in \tau$ is a direct child of $t_{i-1} \in \tau$ $(j = 1, \ldots, k)$, t_1 is the root of the structure τ and t_k is a leaf of the structure.
– A predictive piecewise function $f : \mathbf{X} \rightarrow \mathbf{Y}$, defined according to the hierarchically organized clusters. In particular,

$$\forall u \in \mathbf{X},\ f(u) = \sum_{t_i \in leaves(\tau)} D(u, t_i) f_{t_i}(u) \tag{1}$$

where $D(u, t_i) = \begin{cases} 1 \text{ if } u \in C_i \\ 0 \text{ otherwise} \end{cases}$ and $f_{t_i}(u)$ is a (multi-target) prediction function associated to the leaf t_i.

Clusters are identified according to both the descriptive space and the target space $\mathbf{X} \times \mathbf{Y}$. This is different from what is commonly done in predictive modeling and classical clustering, where only one of the spaces is typically considered. This general formulation of the problem allows us to have the prediction model mining phase which can consider multiple target variables at once. This is the case of predicting the "several" properties of the next event in a case.

The construction of a PCT is not very different from the construction of standard decision tree (see, for example, the C4.5 algorithm [4]): at each internal node t, a test has to be selected according to a given evaluation function. The main difference is that for a PCT, the best test is selected by maximizing the (inter-cluster) variance reduction over the target space $\mathbf{Y}$, defined as:

$$\Delta_Y(C, \mathcal{P}) = Var_{\mathbf{Y}}(C) - \sum_{C_i \in \mathcal{P}} \frac{|C_i|}{|C|} Var_{\mathbf{Y}}(C_i), \tag{2}$$

where C is the cluster associated with t and $\mathcal{P}$ defines the partition $\{C_1, C_2\}$ of C. The partition is defined according to a Boolean test on a predictor variable of the descriptive space $\mathbf{X}$. By maximizing the variance reduction, the cluster homogeneity is maximized, improving at the same time the predictive performance. $Var_{\mathbf{Y}}(C)$ is the variance of $\mathbf{Y}$ in the cluster C. It is computed as the average of variances of target variables $Y_j \in \mathbf{Y}$, that is, $Var_{\mathbf{Y}}(C) = \sum_{Y_j \in \mathbf{Y}} var_{Y_j}(C)$.

2.3 Sliding window model

A sliding window model is the simplest model to consider the recent data of a running case and run queries over the data of the recent past only. Originally

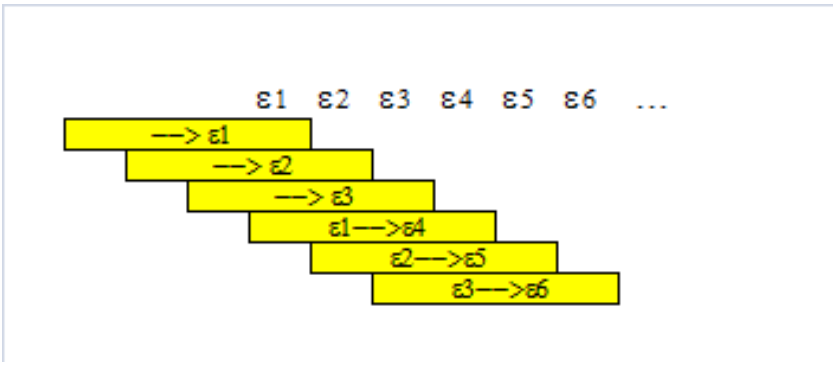


Fig. 1. Sliding window model of a running case with window size $w = 4$.

defined for data stream mining, this type of window is similar to the first-in, first-out data structure. When the event ϵ_i is acquired and inserted in the window, the latest event ϵ_{i-w} is discarded (see Figure 1). w is the size of the window.

Definition 3 (Sliding window model). *Let w be the window size of the model. A sliding window model views a case $\mathcal{C}$ as a sequence of overlapping windows of events,*

$$\mathcal{C}(1 \to w),\ \mathcal{C}(2 \to w+1),\ \ldots,\ \mathcal{C}(i-w+1 \to i),\ \ldots, \tag{3}$$

where $\mathcal{C}(i-w+1 \to i)$ is the series of the w events ϵ_{i-w+1}, ϵ_{i-w+2}, ..., ϵ_{i-1}, ϵ_i of the case $\mathcal{C}$ with $time(\epsilon_{i-j-1}) \leq time(\epsilon_{i-j})$ (for all $j = 0, \ldots, w-2$).

By considering the sliding window model, the window $\mathcal{C}(i-w+1 \to i)$ defines the recent history of the event ϵ_i.

3 Framework for PCT-based Event Forecasting

We use the sliding window model to transform our event-based forecasting problem into a predictive clustering problem where the target variables are the properties of the next event in the case. A PCT is learned off-line from an event log which records full cases and used on-line to forecast the next event expected in a running case (see Figure 2).

Let $\mathcal{L}$ be the event log which records full cases of a process $\mathcal{P}$. In the off-line phase, $\mathcal{L}$ is transformed in a training set $\mathcal{T}$. This transformation is performed by using the sliding window model. Let w be the window size. Each training case $\mathcal{C} \in \mathcal{L}$ is transformed in a bag $training(\mathcal{C}, w)$ of training examples. This bag collects a training example for each event ϵ_i of $\mathcal{C}$ so that the training example $\mathbf{xy}(\mathcal{C}(i-w+1 \to i))$ is generated based upon the sliding window history of the event. Formally,

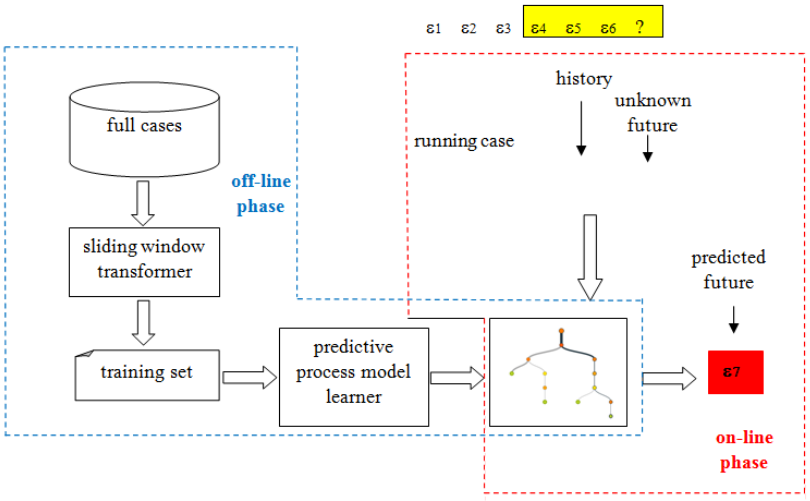


Fig. 2. Event forecasting framework. In the off-line phase, the sliding window model is used to transform the event forecasting task in a predictive clustering task: a PCT is learned from historical full traces of a process. In the on-line phase, the future event of a running case is forecast from its known past.

$$training(\mathcal{C}, w) = \{\mathbf{xy}(\mathcal{C}(i - w + 1 \rightarrow i)) | i = 1, \ldots, |\mathcal{C}|\}, \tag{4}$$

where $|\mathcal{C}|$ denotes the length of the case $\mathcal{C}$, and

$$\mathcal{T} = \bigcup_{\mathcal{C} \in \mathcal{T}} training(\mathcal{C}, w). \tag{5}$$

Each property of an event is transformed into a variable. The target space $\mathbf{Y}$ is populated with variables originated from the newest event in the sliding window, while the descriptive space $\mathbf{X}$ is populated with variables generated from the oldest $w - 1$ time-delayed events in the sliding window. The window size influences the size of the descriptive space $\mathbf{X}$. The longer the window history, the higher the number of the descriptive variables considered for the predictive clustering task. The timestamp is used when generating the descriptive space only. It is transformed into the time (in seconds) gone by the beginning of the case. When an optional property lacks in the related event, the associated variable assumes the value "none" in the training example. An example of this sliding window transformation of a case is reported in Example 1.

Example 1 (Sliding window transformation). Let us consider the case 1 of the event log reported in Table 1. The sliding window transformation of this case generates nine training examples, one for each event in the case. Each timestamp is transformed into the time (in seconds) gone by the beginning of the case.

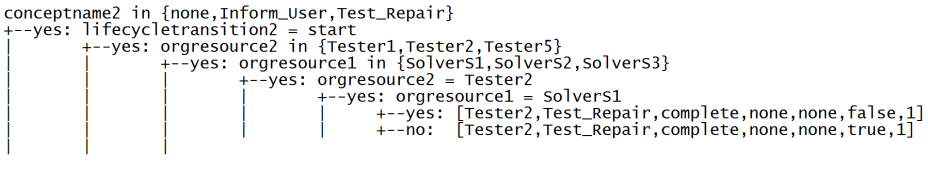


Fig. 3. A fragment of PCT learned from the event log *repair*.

By considering the window size $w = 3$, the following training examples are generated:

none, 0,none, none, none, none, none, none, none,
(1) none, 0,none, none, none, none, none, none, none,
Register, System, complete,none, none, none, none
none, 0,none, none, none, none, none, none, none,
(2) Register, 0, System, complete,none, none, none, none,
Analyze Defect, Tester3, start, none, none, none, none
Register,0,System, complete,none, none, none, none,
(3) Analyze Defect,0,Tester3,start,none, none, none, none,
Analyze Defect, Tester3, complete, T2, 6, none, none
. . .
TestRepair, 1320, Test3, complete, none, none, none, none,
(9) Inform User, 2820, System, complete, none, none, none, none,
Active Repair, System, complete, none, none, true,0

In this case, the descriptive variables are generated from the properties of the oldest two $(w-1)$ time-delayed events in the window, while the target variables (in italics) are generated from the properties of the last event of the window.

Let τ be the PCT learned from $\mathcal{T}$. It generalizes an event-based predictive process model for the process $\mathcal{P}$ (see Figure 3). In the on-line phase, this process model can be used to forecast properties of the next event in each new running case of $\mathcal{P}$. The forecast is that produced by traversing τ based on the properties of the past $w-1$ time delayed events in the case. The selected leaf contains predictions of properties for next event.

4 Empirical Study

The event-based forecasting approach described in this paper processes event logs in the XES[1] format, that is, the XML-based standard for event logs. The predictive clustering tree learner is CLUS[2] [2]. We demonstrate the applicability of the proposed approach by using three benchmark event logs[3].

[1] http://www.xes-standard.org/
[2] http://dtai.cs.kuleuven.be/clus/
[3] http://www.processmining.org/event_logs_and_models_used_in_book

4.1 Event log description

We consider event logs recorded for three different processes. These logs contain the mandatory information about activity, resource, lifecycle and time as well as process-specific optional attributes.

The event log *reviewing* handles reviews for a journal. It consists of 100 cases (papers) and 3730 events. Each paper is sent to three different reviewers. The reviewers are invited to write a report. However, reviewers often do not respond. As a result, it is not always possible to make a decision after a first round of reviewing. If there are not enough reports, then additional reviewers are invited. This process is repeated until a final decision can be made (accept or reject). The optional properties are Result by Reviewer A (accept, reject), Result by Reviewer B (accept, reject), Result by Reviewer C (accept, reject), Result by Reviewer X (accept, reject), accepts (0, 1, 2, 3, 4, 5) and rejects (0, 1, 2, 3, 4, 5). The case length ranges between 11 and 92.

The event log *repair* is about a process to repair telephones in a company. It consists of 1104 cases and 11855 events. The process starts by registering a telephone device sent by a customer. After registration, the telephone is sent to the Problem Detection Department where the defect is analyzed and classified. Once the problem is identified, the telephone is sent to the Repair Department which has two teams: one of the team fixes simple defects and the other fixes the complex defects. Once the repair employer finishes, the device is sent to the quality Assurance department. If the telephone is not repaired, it is sent again to the Repair department. Otherwise the case is archived and the telephone is sent to the customer. The company tries to fix a defect a limited number of times. The optional properties are defect type (ten categories), phone type (three categories), defect fixed (true or false) and number of repairs(0, 1, 2, 3). The case length ranges between 4 and 25.

The event log *teleclaim* contains an event log describing the handling of claims in an insurance company. It consists of 3512 cases (claims) and 46138 events. The process deals with the handling of inbound phone calls, whereby different types of insurance claims (household, car, etc.) are lodged over the phone. The process is supported by two separate call centers operating for two different organizational entities. Both centers are similar in terms of incoming call volume and average total call handling time, but different in the way call center agents are deployed. After the initial steps in the call center, the remainder of the process is handled by the back-office of the insurance company. It is noteworthy that this log is difficult to mine; the alpha algorithm fails to extract the right process model [7]. The optional properties are outcome (B insufficient information, S insufficient information, not liable, processed, rejected) and location (Brisbane and Sydney). The case length ranges between 5 and 18.

4.2 Goal and experimental set-up

The goal of this experimental study is to investigate the predictive ability of the process mining approach presented in the paper. The evaluation is performed

on the three event logs described above. The evaluation is performed in terms of several metrics, which include forecasting accuracy, model complexity, and learning time. These performance measures are estimated by using 10-fold cross validation of logs and by varying the window size between two and the maximum length of a case in the log. In particular, the accuracy is measured in terms of the classification accuracy for events of the test running cases. The model complexity is measured in terms of the number of leaves in the learned trees. The computation time is measured in seconds. The experiments were run on an Intel (R) Core(TM) i7-2670QM CPU @ 2.20 GHz server running the Windows 7 Professional.

4.3 Results and discussion

Forecasting accuracy is plotted in Figures 4(a), 4(c), 4(e), model complexity is plotted in Figures 4(b), 4(d), 4(f) 4, while learning time is plotted in Figures 5(a), 5(b), 5(c). Forecasting accuracy is averaged on the target space.

We observe that, for all logs, forecasting accuracy, as well as model complexity and learning time grow up by increasing w. While forecasting accuracy and model complexity metrics reach a (near) stable pick when $w \geq 6$, the time spent to learn a predictive process model grows-up continuously and linearly with the window size. Although the learning time is always below 7 seconds for reviewing and teleclaim cases, 4 seconds for repair cases, our study indicates that accurate forecasts can be produced by focusing the predictive analysis on the five time-delayed past events. This recommends the choice $w = 6$ which, in all logs, guarantees the highest forecasting accuracy with the lowest learning time cost. In addition, we note that, for the choice $w = 6$, the accuracy, averaged on the target space, is always high, above 90% for all logs. This valuable predictive ability of the proposed approach is confirmed by analysing the accuracy metric as it is computed for each target property individually.

Results for $w = 6$ are reported in Table 2. This finer analysis shows that our predictive model is able to forecast the majority of properties of an event with an accuracy that is greater than 92%. Low accuracy is observed only when predicting resource of events of repair cases. A deeper analysis of this process reveals that repair resource domain includes the values SolverC1, SolverC2, SolverC3, SolverS1, SolverS2, SolverS3, System, Tester1, Tester2, Tester3, Tester4, Tester5, Tester6. By grouping SolverC1, SolverC2 and SolverC3 in a SolverC category, SolverS1, SolverS2, SolverS3 in a SolverS category, as well as Tester1, Tester2, Tester3, Tester4, Tester5, Tester6 in a Tester category, the accuracy for forecasting resource passes from to 0.66 to 0.92. This means that the learned model is able, at least, to identify the resource category accurately although it is not very accurate when identifying the individual resource within the specific resource category.

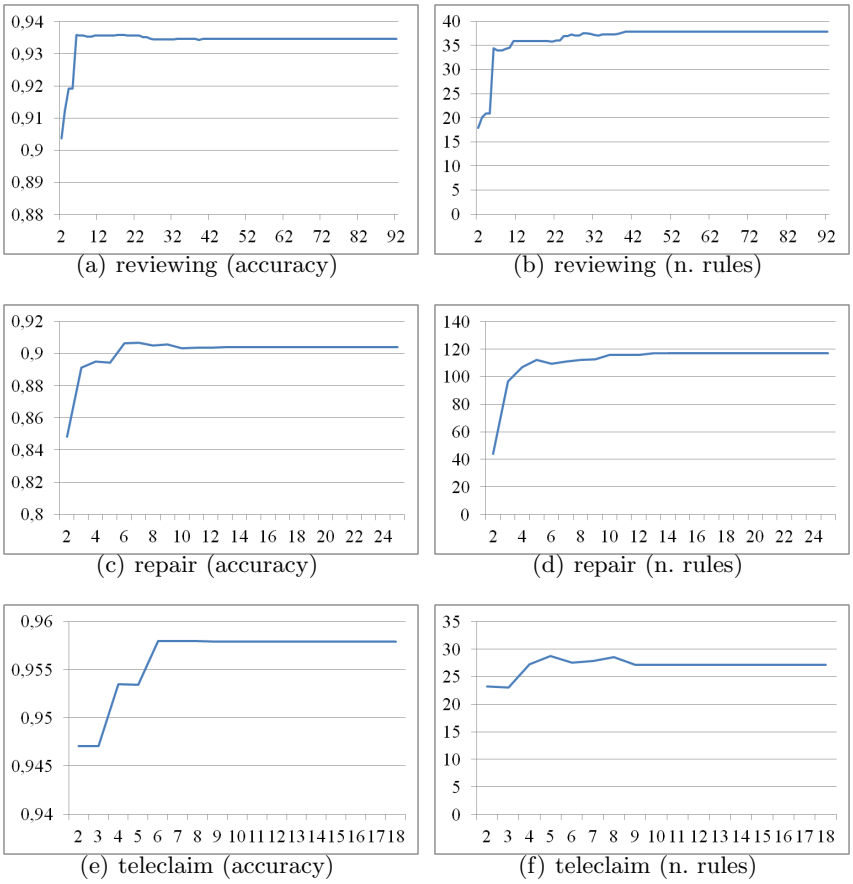
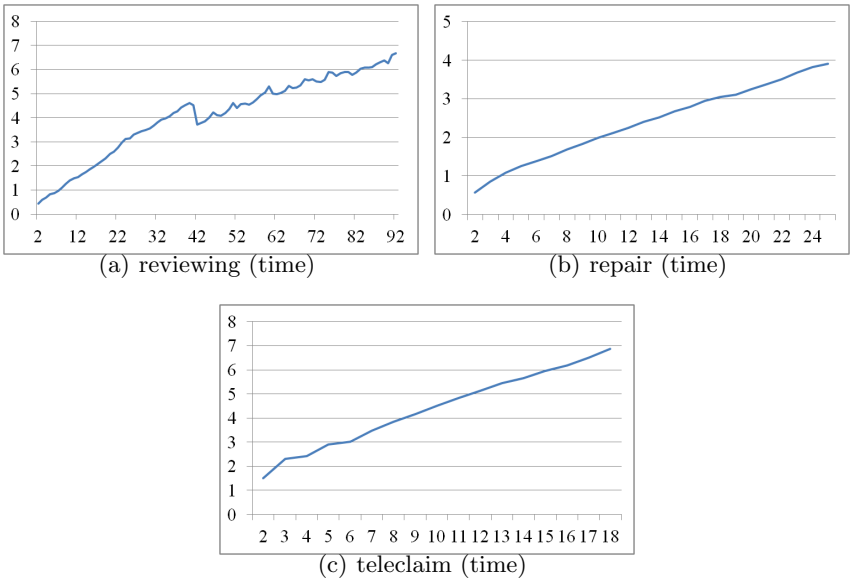


Fig. 4. Forecasting accuracy (4(a), 4(c), 4(e)) averaged on the target space and predictive model complexity (4(b), 4(d), 4(f)).

5 Conclusion

In this paper, we focus on the application of process mining to the prediction of the future of a running case. Given a running case, our prediction allows us answering questions like "what is the activity of the next event?" , "who is the resource triggering the next event?" and so on. We present a data mining framework for event-based prediction support. The framework uses a sliding window-based transformation of the event forecasting task in a predictive clustering task. A predictive process model is learned off-line and used to forecast on-line future events. Forecasts are based on the time delayed events of a running case. In the future, we would like to extend this study by using our forecasts to check conformance of running cases and recommend appropriate actions. We also plan



(a) reviewing (time) (b) repair (time)

(c) teleclaim (time)

Fig. 5. Learning time (seconds).

Table 2. Forecasting accuracy of the predictive process model learned with $w = 6$. The metric is reported for each target attribute.

repair	resource	name	lifecycle	defectType	phoneType	defectFixed	numberRepairs
	0.66	0.92	0.99	0.92	0.93	0.96	0.97

reviewing	resource	name	lifecycle	ResultA	ResultB	ResultC	ResultX	accepts	rejects
	0.79	0.82	1.00	0.99	0.99	0.99	0.93	0.96	0.96

teleclaim	resource	name	lifecycle	outcome	location
	0.96266	0.89786	1.00	0.96666	0.96266

to extend this study by using baseline methods (e.g. single-target classification methods) as well as alternative data stream models (e.g. a landmark [1] based transformation of events as well as relevant event selection [11] for training) in order to be able of accounting for older data together with the newly arriving data when learning the forecasting model.

6 Acknowledgments

This work fulfills the research objectives of the PON 02_00563_3470993 project "VINCENTE - A Virtual collective INtelligenCe ENvironment to develop sustainable Technology Entrepreneurship ecosystems" funded by the Italian Ministry of University and Research (MIUR).

References

1. C. C. Aggarwal, editor. *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer, 2007.
2. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *ICML 1998*, pages 55–63. Morgan Kaufmann, 1998.
3. M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *ACM SIGMOD Record*, 34(2):18–26, 2005.
4. R. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffmann, 1993.
5. F. Takens. Detecting strange attractors in turbulence. *Dynamical systems and turbulence, Warwick*, (898(1)):366–381, 1981.
6. W. van der Aalst, A. Adriansyah, A. Medeiros, F. Arcieri, T. Baier, T. Blickle, J. Bose, P. Brand, R. Brandtjen, J. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. Leoni, P. Delias, B. Dongen, M. Dumas, S. Dustdar, D. Fahland, D. Ferreira, W. Gaaloul, F. Geffen, S. Goel, C. Gunther, A. Guzzo, P. Harmon, A. Hofstede, J. Hoogland, J. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. Rosa, F. Maggi, D. Malerba, R. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. Motahari-Nezhad, M. Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. Seguel Prez, R. Seguel Perez, M. Sepulveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, and M. Wynn. Process mining manifesto. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. Springer Berlin Heidelberg, 2012.
7. W. M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
8. W. M. P. van der Aalst, M. Pesic, and M. Song. Beyond process mining: from the past to present and future. In *the 22nd international conference on Advanced information systems engineering*, CAiSE'10, pages 38–52, Berlin, Heidelberg, 2010. Springer-Verlag.
9. W. M. P. van der Aalst, M. H. Schonenberg, and M. Song. Time prediction based on process mining. *Inf. Syst.*, 36(2):450–475, Apr. 2011.
10. H. M. W. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. Xes, xesame, and prom 6. In P. Soffer and E. Proper, editors, *Information Systems Evolution - CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers*, volume 72 of *Lecture Notes in Business Information Processing*, pages 60–75. Springer, 2010.
11. I. Zliobaite. Combining time and space similarity for small size learning under concept drift. In J. Rauch, Z. W. Ras, P. Berka, and T. Elomaa, editors, *Proceedings of the 18th International Symposium on Foundations of Intelligent Systems, ISMIS 2009*, volume 5722 of *Lecture Notes in Computer Science*, pages 412–421. Springer, 2009.