Predictive Process Monitoring in Apromore

Ilya Verenich[1,2], Stanislav Mõškovski[2], Simon Raboczi[3], Marlon Dumas[2], Marcello La Rosa[3], and Fabrizio Maria Maggi[2]

[1] Queensland University of Technology, Australia
ilya.verenich@qut.edu.au
[2] University of Tartu, Estonia
{stanislav.myshkovski,marlon.dumas,f.m.maggi}@ut.ee
[3] University of Melbourne, Australia
{simon.raboczi,marcello.larosa}@unimelb.edu.au

Abstract. This paper discusses the integration of Nirdizati, a tool for predictive process monitoring, into the Web-based process analytics platform Apromore. Through this integration, Apromore's users can use event logs stored in the Apromore repository to train a range of predictive models, and later use the trained models to predict various performance indicators of running process cases from a live event stream. For example, one can predict the remaining time or the next events until case completion, the case outcome, or the violation of compliance rules or internal policies. The predictions can be presented graphically via a dashboard that offers multiple visualization options, including a range of summary statistics about ongoing and past process cases. They can also be exported into CSV for periodic reporting or to be visualized in third-parties business intelligence tools. Based on these predictions, operations managers may identify potential issues early on, and take remedial actions in a timely fashion, e.g. reallocating resources from one case onto another to avoid that the case runs overtime.

Keywords: Process mining, predictive monitoring, business process, machine learning

1 Introduction

Predictive Process Monitoring is an emerging paradigm based on the continuous generation of predictions about the future values of user-specified performance indicators of a currently running process execution [4]. In this paradigm, a user defines the type of predictions they are interested in and provides a set of historical execution traces. Based on the analysis of these traces, the idea of predictive monitoring is to continuously provide the user with predictions and estimated values of the performance indicators. Such predictions generally depend both on: (i) the sequence of activities executed in a given case; and (ii) the values of data attributes after each activity execution in the case.

As an example, consider a business process for managing loan applications; the applicant can be advised about the combinations of loan amount and duration that are the most likely to lead to an acceptance of the application, given the history of the application until now and the personal data of the applicant (e.g., age, salary, etc.).

A range of approaches have been proposed in the literature to tackle common predictive process monitoring tasks [2]. However, these approaches have largely remained in the academic domain and have not been widely applied in real-time scenarios where users require a continuous predictive support.

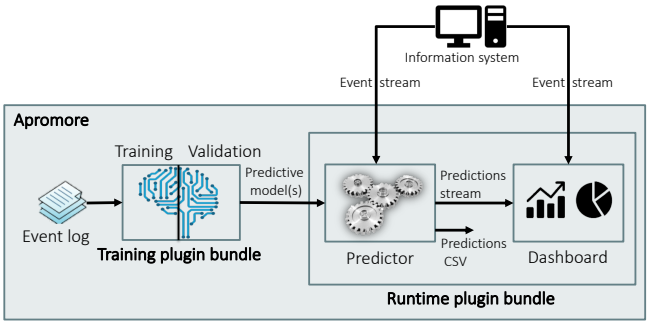

Fig. 1. High-level architecture of the predictive monitoring functionality of Apromore.

To fill this gap, last year we started the *Nirdizati* open-source project, aimed at developing a Web-based tool for the predictive monitoring of business processes. In this paper, we document the developments of Nirdizati since its creation, and its integration into the process analytics platform Apromore.

The two core components of Nirdizati, namely *Training* and *Runtime*, have been integrated as two bundles (i.e. sets) of plugins into Apromore (Fig. 1). The Training plugin bundle takes as input a business process event log stored in the Apromore repository, and produces one or more predictive models, which can then be deployed to the runtime predictive monitoring environment. Once a model is deployed, the Runtime plugin bundle listens to a stream of events coming from an information system supporting the process, or produced by replaying an event log stored in the repository, and creates a stream of predictions. These predictions can then be visualized in a Web dashboard or exported into a CSV (comma-separated value) file for periodic reporting and to be used within third-party business intelligence tools.

2 Apromore Platform

Apromore is a Web-based advanced process analytics platform, developed by the BPM community under an open-source initiative.[1] Apromore was originally conceived as an advanced process model repository. However, today it offers a wide range of features which go beyond those for managing large process model collections, and include a variety of state-of-the-art process mining techniques. These are techniques for the automated discovery of BPMN models, for the conformance checking of BPMN models against event logs, the replaying of event logs on top of BPMN models, the detection and characterization of process drifts from event logs, the visual analysis of process performance, and many others.

All these features are exposed through a Web portal, and organized according to the phases of the BPM lifecycle: discovery, analysis, redesign, implementation and monitoring. These features can also be accessed as external Web services by third-party BPM software environments, such as ProM (for process mining) and WoPeD (for process modeling and verification).

From a technology viewpoint, Apromore relies on four core technologies: Spring, ZK, OSGi and Eclipse Virgo. Spring provides a simplified management of Java-based

[1] `http://apromore.org`

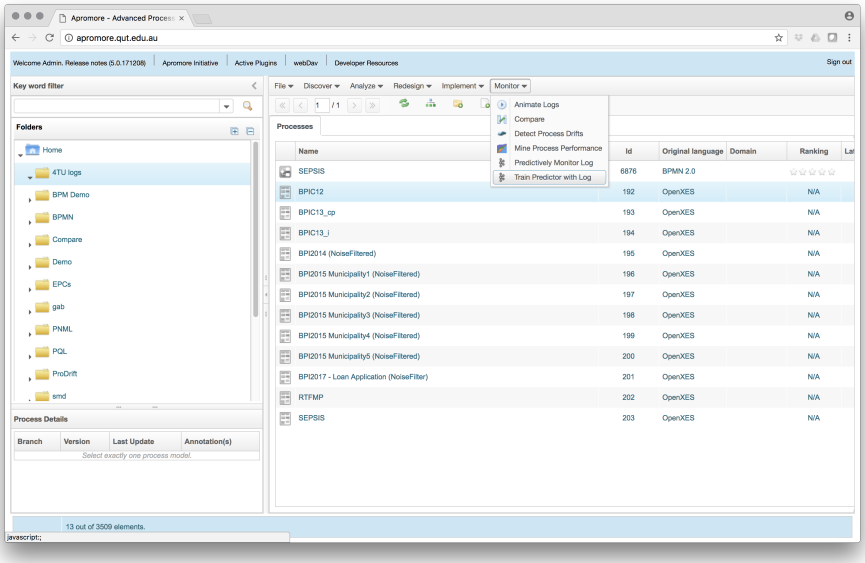

Fig. 2. Apromore's Portal with predictive monitoring functionality highlighted.

enterprise applications through the use of Java annotations and XML configurations. ZK is an AJAX framework used for Apromore's main Web interface (the *Portal*). OSGi provides a flexible framework for managing component dependencies through plugin bundles. Finally, Eclipse Virgo is a Web server based on the OSGi component model.

To equip Apromore with predictive process monitoring capabilities, we have wrapped the two core components of Nirdizati into two OSGi plugin bundles for Apromore: Training and Runtime. Each bundle is a set of OSGi plugins which encapsulate the logic or the user interface (UI) of the various functions offered by Nirdizati. For example, the runtime predictor is a logic plugin, while the runtime dashboard is a portal plugin (UI). These two bundles are accessible from the Monitoring menu of the Apromore Portal (see Figure 2). One can select an event log stored in the repository, and use it to train, tune and test a variety of predictive models, by launching the training plugin bundle. Next, the runtime bundle can be used to stream an event log from the repository, or hook into a live external stream, in order to generate predictions as process cases unfold.

In the next sections, we introduce a working example and use this to describe the functionality of the training and runtime plugins in detail.

3 Running Example

As a running example, throughout this paper we will consider a purchase-to-pay process of an IT vendor. The process starts with lodging a purchase order and ends when requested goods have been supplied.

For use with the Training plugin bundle, we extracted an event log of completed purchased orders, while ongoing orders were fed into the Runtime plugin bundle to make predictions for them. The corresponding event log contains a number of case attributes, such as the total cost of the order, supplier identifier, supplier location, delivery

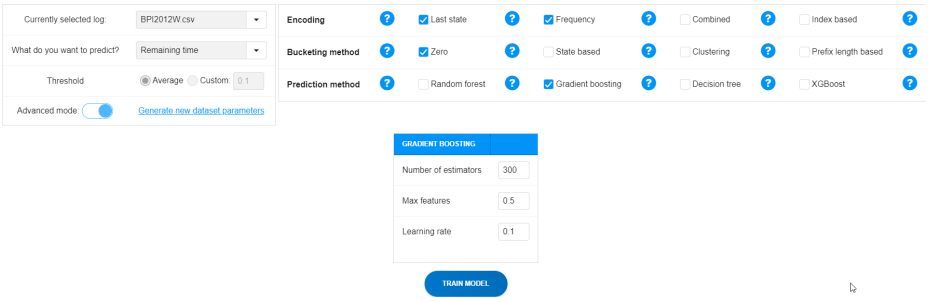

Fig. 3. Training configuration screen.

type and the target supply date. Moreover, each event contains identifier of an employee who performed it.

In this process, stakeholders are interested in predicting four variables:

- *Late supply*. It is a boolean variable indicating whether or not the case will be closed before the supply date of that case.
- *Delay Rank* indicating the degree of potential delay, namely “Just in case”, “Mild”, “Moderate”, “Severe”.
- *Next activity* indicating which activity will be performed right after the current one.
- *Remaining time* until case completion.

In order to make more accurate predictions, we performed some basic feature engineering before feeding the log into Apromore. For example, to take into account resource contention, we added the number of currently open cases as an event attribute. Furthermore, instead of the absolute target supply date, we used the number of days relative to the case start date. Intuitively, if this number is lower, there exists a higher probability of missing the deadline (Late Supply).

4 Training Plugin Bundle

The Training plugin bundle provides several algorithms for generating predictive models suitable for different types of predictions. Specifically, it is able to build models for predicting remaining time, the next activity to be performed, whether a case will exceed a specified duration threshold, as well as various static case attributes, for example, the total cost of the order. To this aim, the Training bundle involves two phases: a training and a validation phase. In the former, one or more predictive models are fitted; in the latter, their suitability to the specific dataset is evaluated, so as to support the user in selecting the predictive model that ensures the best results.

The Training bundle is composed of a front-end application (Fig. 3), which allows users to select the prediction methods and to assess the goodness-of-fit of the built models, and a back-end application for the actual training and validation. From the data flow perspective, the back-end application performs several tasks shown in Fig. 4.

Firstly, when a user uploads their log, the tool extracts and categorizes data attributes of the log. In order to properly construct feature vectors from business process traces, the attributes need to be categorized into static case attributes and dynamic event attributes. To this end, for each attribute in the log, we check whether it changes through-

out the case lifetime. On the other hand, each attribute needs to be designated as either numeric or categorical.

In order to that, we count the number of unique levels in the values of each attribute and if it less than a certain threshold, we decide the attribute to be categorical, otherwise it is numeric. These procedures are performed automatically upon the log uploading. Nevertheless, the user is given an option to override the automatic attribute definitions. The resulting definitions are saved in a configuration file in a JSON format (Fig. 5). Secondly, the log is internally split into training and validation set in a 80-20 proportion. The former is used to train the model, while the latter is used to evaluate the predictive power of the model. Next, all traces of a business process need to be represented as fixed-size feature vectors in order to train a predictive model. To this end, several encoding techniques were proposed in [1] and further refined in [3], out of which we support four, namely last state encoding, frequency (aggregation) encoding, combined encoding and lossless index-based encoding.

While some of existing predictive process monitoring approaches train a single classifier on the whole event log, others employ a multi-classifier approach by dividing the prefix traces in the historical log into several buckets and fitting a separate classifier for each such bucket. At run-time, the most suitable bucket for the ongoing case is determined and the respective classifier is applied to make a prediction. Various bucketing types have been proposed and described in detail in [3]. The Training bundle supports four types of bucketing: zero bucketing (i.e. fitting a single classifier), state-based bucketing, clustering-based bucketing and prefix length-based bucketing.

For each bucket of feature vectors, we train a predictive model using one of four supported machine learning techniques: decision tree, random forest, gradient boosting and extreme gradient boosting (XGBoost). For each technique, a user may manually enter the values of the most important hyperparameters. For example, when fitting a gradient boosting model, a user may choose the number of weak learners (trees), the number of features to be used for each split and the learning rate.

In order to accommodate users with varying degrees of expertise in machine learning and predictive process monitoring, the plugin bundle offers two training modes – basic and advanced. By default, the basic mode is activated wherein a user only needs to choose the log and prediction target. If the prediction target is based on the logi-

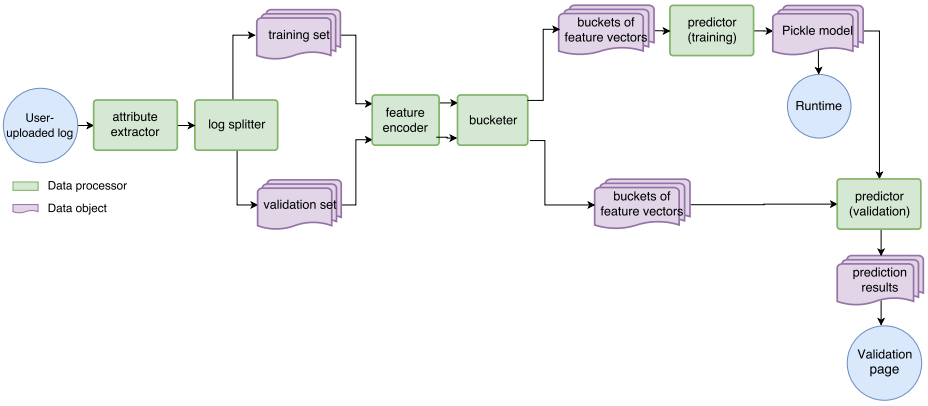

Fig. 4. High-level data flow diagram of the Training plugin bundle.

```
{
"case_id_col":"Line_ID",
"activity_col":"Activity_Name",
"timestamp_col":"Activity_End_Time",
"static_cat_cols":["Supplier_ID","Supplier_Location","
    Delivery_Type","Sector"],
"dynamic_cat_cols":["Employee_ID","weekday"],
"static_num_cols":["Line_Total_Cost","Target_Supply_Date_delta"]
    ,
"dynamic_num_cols":["elapsed","timesincelastevent","open_cases"]
    ,
"ignore":["Activity_Start_Time","hour","Milestone","Key","
    Target_Supply_Date","Part_ID"],
"future_values":["Late_Supply","Delay_Rank"],
}
```

Fig. 5. Example training configuration file.

cal rule – whether the case duration will exceed the specified threshold, a user is also invited to key in the threshold value. For all the other settings – bucketing method, encoding method and prediction method and its hyperparameters – the default values which usually achieve the best prediction accuracy will be used. Experienced users may switch the advanced mode toggle and manually choose bucketing, encoding and prediction method settings or any sensible combination thereof. The latter is especially useful when a user wants to train and compare multiple models, e.g. using various sequence encoding methods.

The status of the trained models can be verified using the collapsible drawer in the right-hand corner. Upon the training completion, a serialized Python object in the pickle format is produced. It describes a trained predictive model and which includes:

- Configuration parameters of the predictors (whether it is a classifier or a regressor, what learning algorithm it uses)
- Definition of each column of the event log (static or dynamic, numeric or categorical). This information allows the Runtime plugin bundle to construct a feature vector from a given partial trace.
- The bucketing function, which given an input sample, allows us to determine which classifier/regressor should be used. Unless a zero bucketing is used, the pickle file has a different classifier/regressor per bucket.
- For each bucket, the trained model, ready to be taken as input by the selected classification algorithm, e.g. in the case of decision trees, the whole tree representation.

The predictive power of the trained model(s) can be evaluated on a held-out validation set. By default, a user will see the average accuracy across all partial traces after a certain number of events have completed. This evaluation method was also used in [1] and [3]. For classification tasks (e.g. prediction of Late Supply and Delay Rank), a user can choose which metrics to plot among accuracy score, F1 score and logarithmic loss. For regression tasks (e.g. remaining time), a user can choose between mean absolute error, either raw or normalized, and root mean square error. The accuracy of a particular model can be visually compared with that of other models trained for the same log and the same prediction target (Figure 6). Additionally, one can check a scatterplot of predicted vs. actual values (for regression tasks) or a confusion matrix (for classification tasks) and assess the relative importance of each feature for the chosen predictor.

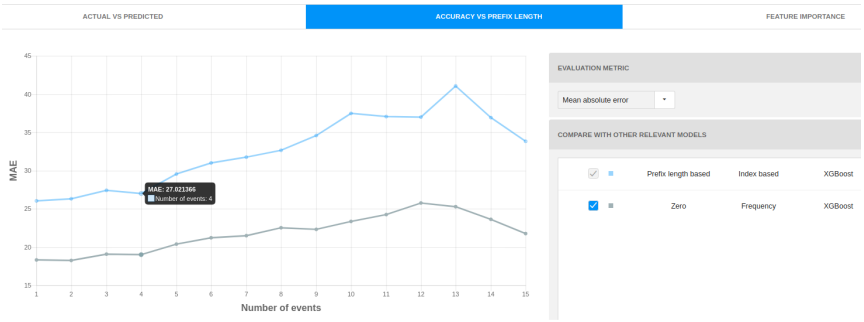

Fig. 6. Model validation page of the Training plugin bundle.

5 Runtime Plugin Bundle

Once the predictive models have been created, they can be deployed to the Runtime predictive monitoring environment of Apromore, to make predictions on ongoing cases. The Runtime plugin bundle can be used to stream an event log from the repository, or hook into an external stream. Either way, the input stream is transformed into a stream of predictions which is visualized in a Web-based dashboard. The transformation is implemented using the dataflow pipeline in Fig. 7.

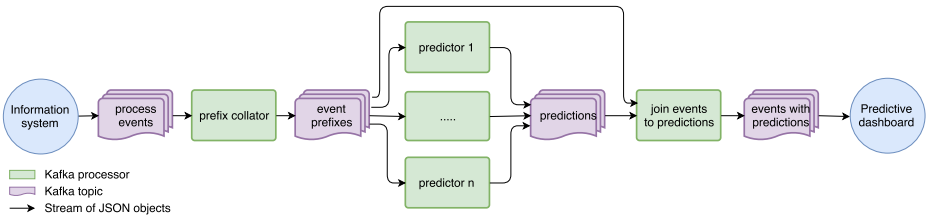

Fig. 7. High-level data flow diagram of the Runtime plugin bundle.

The pipeline is built on top of the open-source Apache Kafka stream processing platform. The "predictor" components of the pipeline are the predictive models from the Training plugin bundle. The "topic" components are network-accessible queues of JSON messages with publisher/subscriber support. This allows the computationally intense work of the predictors to be distributed across a cluster of networked computers, providing scalability and fault-tolerance. The "collator" component accumulates the sequence of events-to-date for each case, such that the prediction is a stateless function of the trained predictive model and of the case history. This statelessness is what allows the predictors to be freely duplicated and distributed. The "joiner" component composes the original events with the various predictions, ready for display on the dashboard.

The dashboard provides a list of both currently ongoing cases (colored in gray) as well as completed cases (colored in green), as shown in Fig. 8. For each case, it is also possible to visualize a range of summary statistics including the number of events in the case, its starting time and the time when the latest event in the case has occurred. For the ongoing cases, the Runtime plugin bundle provides the predicted values of the performance indicators the user wants to predict. For completed cases, instead, it shows

the actual values of the indicators. In addition to the table view, the dashboard offers other visualization options, such as pie charts for case outcomes and bar charts for case durations. It is also possible to export the predictions in a CSV file, for periodic reporting and for importing into third-party business intelligence tools.

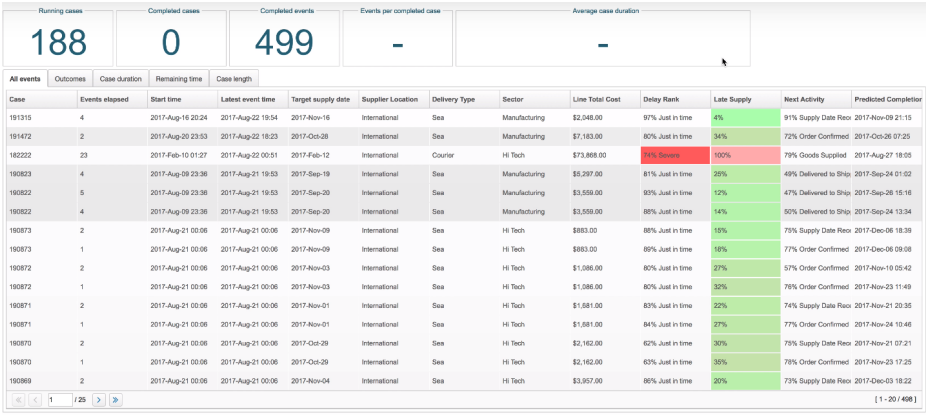

Fig. 8. Main view of the dashboard in the Runtime plugin bundle.

Process workers and operational managers can set some process performance targets and subscribe to a stream of warnings and alerts generated whenever these targets are predicted to be violated. Thus, they will be capable of making informed, data-driven decisions to get a better control of the process executions. This is especially beneficial for processes where process participants have more leeway to make corrective actions (for example, in a lead management process).

6 Conclusion

Through the integration with Nirdizati, Apromore offers a configurable full-stack Web tool that supports users in selecting and tuning various prediction models, and that enables the continuous prediction of different process performance indicators at runtime. Predictions can be presented visually in a dashboard or exported for periodic reporting.

Video demos of the model training and of the runtime functionality can be found at `http://youtu.be/CYwLTKAbF1Q` and at `http://youtu.be/Q4WVebqJzUI`. The source code is available under the LGPL version 3.0 license at `https://github.com/apromore/ApromoreCode`.

References

1. Leontjeva, A., Conforti, R., Francescomarino, C.D., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. BPM (2015)
2. Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortes, A.: Predictive monitoring of business processes: a survey. IEEE TTSC (2017)
3. Teinemaa, I., Dumas, M., La Rosa, M., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. CoRR abs/1707.06766 (2017)
4. Verenich, I.: A general framework for predictive business process monitoring. CAiSE Doctoral Consortium (2016)