UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Kerwin Jorbina

A Web-Based Tool For Predictive Process Analytics

Master's Thesis (30 ECTS)

Supervisor: Fabrizio Maria Maggi, PhD
Supervisor: Chiara Di Francescomarino, PhD
Supervisor: Chiara Ghidini, PhD

Tartu 2017

A Web-Based Tool For Predictive Process Analytics

Abstract:

Predictive Process Monitoring aims at exploiting event logs in business processes by providing predictions and forecasts on key business metrics such as time, cost and activity executions. As the interest in this field grows, various methods and approaches have been implemented in both academia and industry sectors in order to produce visual results that are understandable to the users. In this Master's Thesis, we propose a web-based framework and tool that enables participants in this field to build quick visualizations on their predictive models for evaluation. Furthermore, this project intends to have an independent front-end application which can work with any method running on the back-end as a web-service that is used in the prediction process. Finally, this project looks into the realm of inter-case predictions which uses multiple cases in building a prediction model of an event log.

Keywords:
Predictive Process Monitoring, Process Mining, Process Analytics Tool

CERCS: P170- Computer Science, Numerical Analysis, Systems, Control

Veebipõhine tööriist Ennustuspõhiseks Protsessianalüütikaks **Lühikokkuvõte:**

Äripotsesside ennustav seire kasutab äriprotsesse toetavaid sündmuste logisid, luues ennustusi äriindikaatorite suhtes, näiteks aeg, maksumus ja järgnevad sündmused. Arvestades kasvavat huvi valdkonna vastu, on nii äri- kui akadeemilises valdkonnas loodud mitmeid lahendusi visualiseerimaks neid ennustusi kasutaja jaoks arusaadaval kujul. Käesolevas magistritöös pakume välja veebipõhise raamistiku ja tööriista, mis lubab kasutajatel hindamiseks kiirelt visualiseerida ennustusmudeleid. Veelgi enam, selle töö käigus loodi veebirakendus, mis suudab suhelda veebiteenuste abil ükskõik millise ennustusprotsessis kasutatava serveri rakendusega. Lõpuks uurib antud töö ka mitme juhtumi põhist ennustust, mis tähendab, et mudelid luuakse kasutades mitut sündmuste logi juhtumit.

Võtmesõnad:
Protsessi Ennustav Seire, Protsessikaeve, Protsessi Analüüsitööriistad

CERCS: P170- Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Predictive Process Analytics (PPA) is the art of visualizing the results in the evaluation phase of prediction methods. In this section, we discuss the concepts behind process prediction, the current issues encountered and the goals of this project.

1.1 Motivation

Exploitation of diverse types of data in the modern era allows one to gain an advantage in business ventures. Most advanced companies store information about process executions into textual files. These are called *event logs*, which are storages for the historical traces depicting completed process executions. Process mining [21], a spin-off field in Business Process Management deals with this undertaking where it desires to show the flow of activities or how it plays (process discovery), to see if the execution is conformant when performing a replay against an existing business process model (conformance checking) and to improve or extend a process model with the use of the information from the log (process enhancement). Furthermore, it deals with operational support, which is an online technique that uses pre-mortem event data, i.e. ongoing cases that can be analyzed at runtime.

There are three main goals of operational support. First is to detect any deviant cases, recently expired deadlines and anything that goes wrong in the process. Second is to predict when cases will finish or if the outcome is desirable or not. Lastly, it intends to recommend future activities and its respective performers. The goal of predictive process monitoring is to use historical traces stored in event logs to derive a predictive model that will yield practical results helping eliminate risks of failure of an ongoing case, predicting its outcome and its time of completion as well as its compliance with respect to a process model.

Take for example a purchase done in the online delivery process. The goal of the supplier is to send the goods to the procurer on the scheduled time arbitrated. A normal flow of activities may include collecting the item, storing it inside the box, providing it to a courier, giving the item over to the customer, and validating whether the delivery is successful. These executions are logged together with other optional or mandatory details like location, performers of tasks and conditions during the delivery. The goal of predictive monitoring is to use these previous executions to find underlying patterns and see if the newly executed process instances will result in a successful and timely delivery.

However, current approaches hardly take into consideration dependencies among different historical cases. Furthermore, the previously performed methods don't offer a visualization platform for others to easily re-use on their own datasets. This project would like to collate those approaches and provide a tool where users may combine different techniques. Furthermore, this kind of tool allows concerned parties to easily conduct experiments using previous approaches, thus giving them usable baseline results for their own research. In addition, we provide a method to take both intra and inter case features into consideration for prediction.

1.2 Goal and Problem Statement

In this thesis, we combine and present the different methods used in predictive process monitoring for us to provide an analytics framework that can be re-used by interested participants for the faster visualization of evaluation results. This thesis mainly aims at simplifying the evaluation process in process predictions by providing a base tool and web services performing the predictive algorithms.

We define our research goals as follows:

1. How can we categorize prediction methods from existing approaches?
2. How can we support inter-case feature encoding for predictive monitoring?
3. How can we design a web-based framework for supporting such prediction methods?

In addressing these questions, we look into papers that tackled process predictions. We look into their approaches, specifically looking into the type of prediction and what methods they use to achieve it.

This paper is structured as follows:

Section 2 presents the related work done in predictive process monitoring.

Section 3 describes what predictive process monitoring is.

Section 4 presents the main contribution of this project, in particular, we discuss the classification of the types of predictions we implemented in the tool. We also introduce the novel inter-case encoding. In our approach, we implemented it to predict the remaining time of an ongoing case.

Section 5 describes the implementation of the front-end and back-end applications.

Section 6 discusses the evaluation of the new predictive techniques.

Section 7 summarizes the answers to the goals of the project and provide additional input on how one can extend and improve the implementation of the tool.

2 Related Work

This section presents the state of the art in Predictive Process Monitoring. We summarize the approaches by identifying the type of prediction and the method for achieving it.

2.1 Remaining Time Predictions

First, we look at approaches that deal with time predictions, i.e., more in detail, approaches that aim to answer any of the following questions:

1. Given a current state in the process, how much more time do I have to wait for the process to end?
2. Given a current state in the process, how long will it take for me to arrive to this step in the process?

A paper by van Dongen et al. [22] used non-parametric regression for computing the remaining cycle time rather than using the most trivial way, which is to estimate the average cycle time and deduce the elapsed time. Polato et al. [14, 15] instead used different methods for computing the remaining time such as simple regression, regression with contextual information and data-aware transition systems. Ceci et al. [2] also predicted the remaining time by identifying partial process models by sequential pattern mining. These partial process models are used to train and create the predictive models for the estimation of the completion time. Moreover, van der Aalst et al. [20] proposed another approach for time prediction based on annotated transition systems to get the remaining time until completion of a process execution.

2.2 Load Predictions

The goal of this type of prediction is to prepare the organization for making business decisions based on the amount of incoming work. Some questions this type of prediction tries to answer are:

1. How many new cases will be coming in on Monday?
2. How many workers do I need during this time of the season?

The paper of Castellanos et. al. [1] shows the incorporation of time series in their business operations platform to predict aggregated metrics such as average total duration of cases on a given day.

2.3 Outcome Based Predictions

Next, we deal with the type of prediction that classifies the ongoing case. For this type of prediction, we look into categorical results that address the following questions:

1. Will the ongoing trace finish before a time specified?
2. Will the ongoing trace violate service level agreements?
3. Is the ongoing trace in risk of failing?
4. What will be the next activity or sequence of activities after performing a task?

The general goal of this type of prediction is to check for negative outcomes in business processes to carry out reparative actions before the process execution ends.

For example, Polato et al. [15] allow the user to input a threshold time to check if the process will finish before it or not. Verenich et. al. [24] used a temporal deviance criterion by labeling traces as fast or slow based on whether they finish within a specified time threshold. If the trace does not finish for the given time, it is considered as deviant and will be categorized as slow. Federici et al. [7] presented a tool which allows a user to select classification and clustering methods to be able to predict an outcome at a given prefix of an ongoing case. Maggi et al. [12] present an approach that provides an early advice of the outcome.

Conforti et al. [3] presented a recommendation system using Decision Support Systems that guides the process participants in making choices during risky operations. In the prospects of identifying risks of failure, Conforti et al. [4] used a Predictive Risk Monitor where they detect risks as early as possible by using digital sensors and similarities in the business process.

To predict the next task, Polato et al. [15] predict the future sequences of activities given a prefix of the ongoing process instance. They used transition systems to predict the next sequences of tasks. Tax et al. [18] had a different method of predicting sequences of activities using Long Short-Term Memory (LSTM) Neural Networks.

2.4 Cost Predictions

Cost is a key indicator of business processes that organizations are interested to predict. Wynn et. al. [25] implemented an approach to support cost analysis in business processes by using statistics on the historical data in event logs with cost attributes. Their tool allows users to create queries such as:

1. What is the predicted cost of an ongoing case?
2. Who can perform the task the cheapest?

2.5 Intra-case and Inter-case Predictions

Intra-case predictions do not deal with dependencies among traces. Most approaches in predictive process monitoring have focused in this area.

Just recently, a research done by Conforti et. al [4] looked into the inter-case prediction realm which evaluates multiple process executions at one time and present recommendations in terms of who will perform a certain task. They used an algorithm that optimally distributes work items to the resource while minimizing the risk and the overall execution time. In this thesis, we show how to use inter-case features to take into consideration dependencies among traces for prediction.

3 Background

This section describes Predictive Process Monitoring and the methods used in this thesis for the implementation of the analytics tool.

3.1 Predictive Process Monitoring

The main goal of Predictive Process Monitoring (PPM) is to predict the fulfillment of business goals through online monitoring of ongoing process executions. It uses an event log containing historical process executions to build a predictive model able to analyze ongoing traces and make predictions.

Different techniques have been proposed in the context of Predictive Monitoring in Business Processes. They can be classified according to the type of prediction such as time, outcome and conformance of an ongoing trace. For example, one company offers a solution to provide a service that finishes within the same day. To avoid violations in this rule, we can perform predictive process monitoring on running process instances. This is to avoid violations in the contracts and appropriate tasks allocations of resources for the successful completion of a process instance.

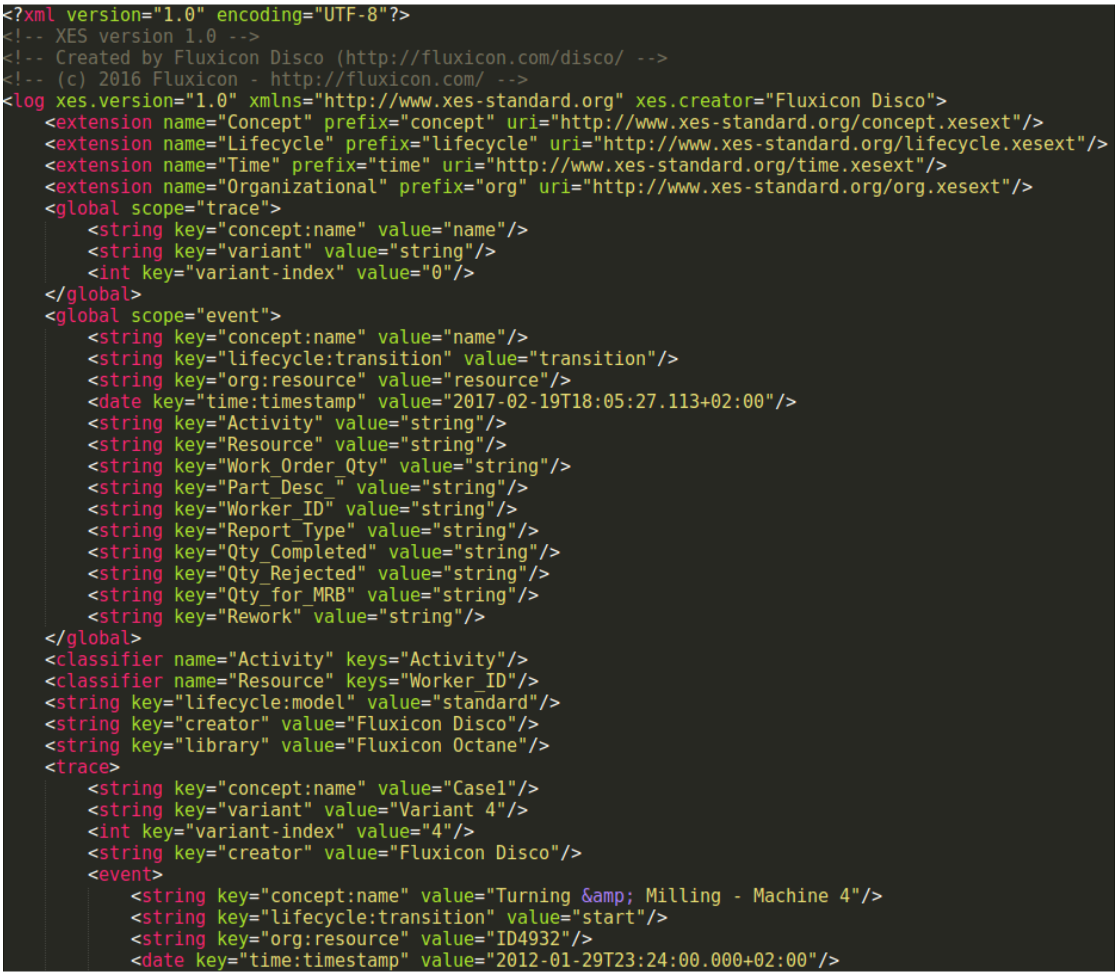
In this subsection, we describe the elements that are needed for predictive process monitoring.

3.1.1 The Event Log

To be able to make an encoded data to create prediction models, we use completed (historical) process executions stored in an event log. In this project, the logs we use are in the Extensible Event Stream (XES) [23] format. An example of the contents of the XES file is shown in Figure 1. It can be seen that this file is following XML specifications where the root is the log which contains the element, trace which further contains events.

In mathematical terms, we let an event log L as the representation of process executions. We let ϵ be the representation of an event log. An event log L is a K-sized sequence of event executions or cases. We define an event log as $L = \{\sigma_i : i = 1, ..., K\}$ where $\sigma_i = (e_i^1, ..., e_i^n) \in \epsilon^*$ is a case i of length n. For predicting an ongoing case, we look into its prefix. We deine the prefix function as $\phi : \epsilon^* \times N^+ \rightarrow \epsilon^*$, which returns the prefix of a case at size n which then gives a sequence $\phi(\sigma_i, n) = (e_i^1, ..., e_i^n) : n \leq n_i$. This function returns σ_i for traces where $n > n_i$. The exteded log $L^* = \{\phi(\sigma_i, n) : \sigma_i \in L, n \leq n_i\}$ is the event log that contans all prefixes.

Moreover, we consider that each event in the sequence has attribute-value (AV) pairs. The AV function is defined as α to be $\alpha : \epsilon \rightarrow A_1 \times ... \times A_p$ with $A_j, j = 1, ..., p$ being p attribute domains. We assume $\exists j : A_j$ as the timestamps TS which can be expressed in UNIX time recordings. In addition, we consider $j = 1, ..., p$ the unknown value $\perp \in A_j$.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- XES version 1.0 -->
<!-- Created by Fluxicon Disco (http://fluxicon.com/disco/ -->
<!-- (c) 2016 Fluxicon - http://fluxicon.com/ -->
<log xes.version="1.0" xmlns="http://www.xes-standard.org" xes.creator="Fluxicon Disco">
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
    <global scope="trace">
        <string key="concept:name" value="name"/>
        <string key="variant" value="string"/>
        <int key="variant-index" value="0"/>
    </global>
    <global scope="event">
        <string key="concept:name" value="name"/>
        <string key="lifecycle:transition" value="transition"/>
        <string key="org:resource" value="resource"/>
        <date key="time:timestamp" value="2017-02-19T18:05:27.113+02:00"/>
        <string key="Activity" value="string"/>
        <string key="Resource" value="string"/>
        <string key="Work_Order_Qty" value="string"/>
        <string key="Part_Desc_" value="string"/>
        <string key="Worker_ID" value="string"/>
        <string key="Report_Type" value="string"/>
        <string key="Qty_Completed" value="string"/>
        <string key="Qty_Rejected" value="string"/>
        <string key="Qty_for_MRB" value="string"/>
        <string key="Rework" value="string"/>
    </global>
    <classifier name="Activity" keys="Activity"/>
    <classifier name="Resource" keys="Worker_ID"/>
    <string key="lifecycle:model" value="standard"/>
    <string key="creator" value="Fluxicon Disco"/>
    <string key="library" value="Fluxicon Octane"/>
    <trace>
        <string key="concept:name" value="Case1"/>
        <string key="variant" value="Variant 4"/>
        <int key="variant-index" value="4"/>
        <string key="creator" value="Fluxicon Disco"/>
        <event>
            <string key="concept:name" value="Turning &amp; Milling - Machine 4"/>
            <string key="lifecycle:transition" value="start"/>
            <string key="org:resource" value="ID4932"/>
            <date key="time:timestamp" value="2012-01-29T23:24:00.000+02:00"/>
```

Figure 1. XES file of the Production Log

Finally, we associate a label for every prefix of σ_i. An example of label is the remaining time $y(\phi(\sigma_i, n)) \in \mathcal{Y}$ where $\mathcal{Y}$ is the domain of the labels. As the case progresses or executing new activities, we take note that the values of these labels also change.

As an example, we can consider the sample XES file in Figure 1. This event log was taken from a production process, where Case 1 is the first event sequence σ_1. Its AV function map the *Turning & Milling - Machine 4* event to "ID4932, 2012-01-29T23:24:00.000+02:00 corresponding to the attributes *resource and timestamp*.The *remaining time* for this event is the difference between its timestamp and the timestamp of the last event in the case. Take note from this example that there is a case attribute *variant* which is static at each event execution in the trace while the *resource* and *timestamp* are dynamic.

3.1.2 Encoding Methods

Encoding is an essential part of data preprocessing. It is required for feeding the algorithms creating the predictive models. In this study, we looked at the encoding methods presented in the paper of Leontejeva et. al. [11]. We used the **Simple Index Based Encoding** in the analytics tool presented in this project.

	consultation	ultrasound	...	payment	label
σ_1	1	1	...	0	false
...					
σ_k	0	0	...	1	true

(a) *boolean* encoding.

	consultation	ultrasound	...	payment	label
σ_1	2	1	...	0	false
...					
σ_k	0	0	...	4	true

(b) *frequency-based* encoding.

	event_1	...	event_m	label
σ_1	consultation		ultrasound	false
...				
σ_k	order rate		payment	true

(c) *simple index* encoding.

	age	event_1	...	event_m	...	department_last	label
σ_1	33	consultation		ultrasound	...	nursing ward	false
...							
σ_k	56	order rate		payment	...	clinic	true

(d) *index latest payload* encoding.

Figure 2. Encoding Methods Introduced by Leontjeva et. al. [11]

This encoding method only takes into account the control-flow of the trace. Every feature in this encoding method is the event from index 0 to n, the specified point or prefix to make the prediction. An example of this encoding and other available methods are shown in Figure 2. These other methods are the following:

1. **Boolean Encoding** assigns a true or false value if an event occurs in the trace.
2. **Frequency Based Encoding** gets the number of occurences of an event in a trace.
3. **Index Latest Payload Encoding** is an extension of simple index encoding but with the inclusion of the data payload at the provided prefix.

For this project, we present two more encoding methods in the Contributions section of the paper.

3.1.3 Prediction Methods

In this subsection, we present the methods in predictions used in this project. We consider these methods as the learning task responsible for building the prediction models.

1. **Classification** In this learning method, the goal is to categorize new observations based on a training set with the same set of observed features.

   (a) **Decision Trees** - A decision tree is a graph in a shape of a tree where there are leaves and branches. Each leaf has a numerical value in terms of probability in order make decisions based on the input variables. The encoded traces will be fed here to be able to create the model that would allow encoded ongoing traces to be predicted.
   (b) **Random Forest** - is a tree ensemble learning method that applies the concept of *bagging* which uses the combination of learning models to increase the accuracy [10]. A given dataset is split into random subsets from which multiple decision trees are created. When doing a prediction, the concept of *voting* is used to produce the final predicted value. This approach in machine learning improves the accuracy and prevents over-fitting of the model.
   (c) K**-Nearest Neighbor (KNN)** - is a classification algorithm that provides a prediction based on the majority population of the K closest neighbors in a given feature space.

2. **Regression** is a machine learning method that predicts a number using a function. Regression methods create a line estimation from a given set of observations. To train the regression models, a dataset in which the last column serves as the target or the expected value is prepared.

   Among the regression approaches, we distinguish:

   (a) **Linear Regression** - allows for detecting functions like

   $$y = f(x)$$

   where *f(x)* is a function that fits a line based on independent variables or the features in the dataset and *y* is the predicted value. If there is only one feature used as the independent variable in the function, it is defined as *Simple*, otherwise, it is considered as *multivariate*.
   (b) **Lasso Regression** - this is the Least Absolute Shrinkage and Selection Operator (Lasso) [10]. This is a regression method that uses a shrinkage method through penalization on large values to prevent it from being weighed more in the prediction.
   (c) **Random Forest Regression** - the concept is similar to the classification approach. The only difference is in the voting mechanism which uses the average results for each tree to get the final value of prediction.
   (d) **XGBoost** - the name of this algorithm is short for *Extreme Gradient Boosting* [8]. This method is nearly the same with Random Forest as it also uses the notion of tree ensembles. They differ though in training where XGBoost builds on weak classifiers (shallow trees which may have two leaves only).

A classifier is added at each time, which improves the previously trained tree ensemble.

3. **Time series** [17] is a series of data recorded at each point in time. This is a discrete-time data which can be plotted on a line chart which gives the ability to show trends, cycles, fluctuations and seasonal movements. An example of a time series models is called **ARMA** (Autoregressive-Moving Average). This univariate model of time series forecasting includes two parts, Autoregressive (AR) and Moving Average (MA). Autoregressive models explain that the current value x_t of the time series can be derived from the data in the past. This can be formulated in the equation:

   $$x_t = c_1 x_{t-1} + c_2 x_{t-2} + ... + c_p x_{t-p}$$

   where c_1, c_2, c_p are constants ($c_p \neq 0$), *p* is the number of steps in the past to look at and *t* as the point in time for the prediction.

   Moving Average model, on the other hand, assumes the white noise errors w_t are combined linearly to produce the output prediction.

   $$x_t = w_t + l_1 w_{t-1} + l_2 w_{t-2} + ... + l_q w_{t-q}$$

   where l_1, l_2, l_p are lags ($l_q \neq 0$) and q as the order.

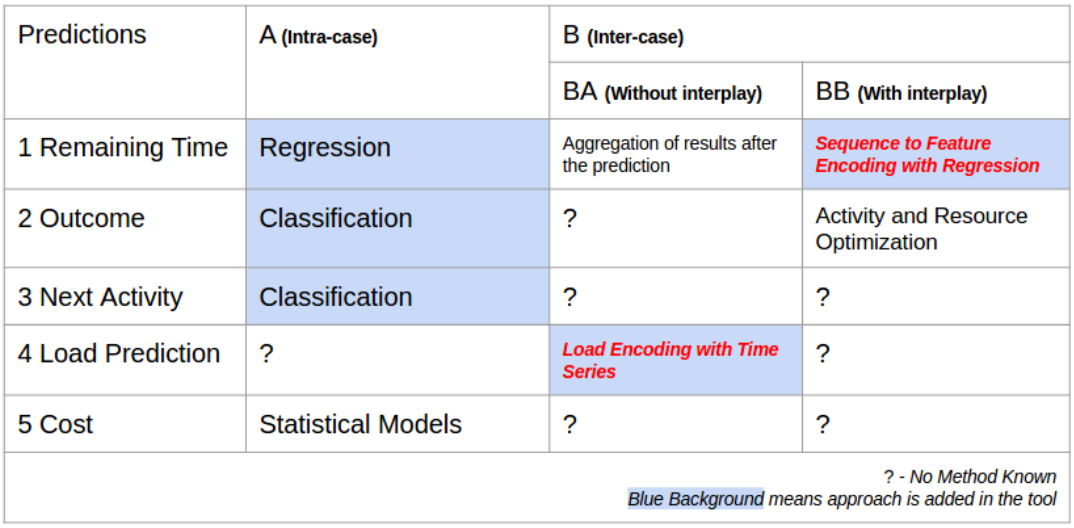
   ARMA is the combination of both equations above which includes both p and q as the autoregressive and moving average terms, respectively.

4 Contribution

We present in this section the answers to the research questions and the architecture of a web-based tool for predictive process analytics.

4.1 Categorization of Prediction Methods in Process Monitoring

We perform a categorization of the prediction methods to find the gaps in the state of the art and select parts which are not yet addressed in Predictive Process Monitoring. This is shown in Table 1.



Predictions	A (Intra-case)	B (Inter-case)	
		BA (Without interplay)	BB (With interplay)
1 Remaining Time	Regression	Aggregation of results after the prediction	***Sequence to Feature Encoding with Regression***
2 Outcome	Classification	?	Activity and Resource Optimization
3 Next Activity	Classification	?	?
4 Load Prediction	?	***Load Encoding with Time Series***	?
5 Cost	Statistical Models	?	?
? - No Method Known *Blue Background means approach is added in the tool*			

Table 1. Method Map in Predictive Process Monitoring

Based on the literature review in Section 2, we present four general types of prediction in process monitoring. To be able to achieve the predictions, we classify the approach as either intra-case or inter-case. In the real world, process executions can happen at the same time, which means that these cases may depend on one another. When predictive process monitoring is done in scenarios where resource contention occurs, it is called *dependent inter-case predictive monitoring or inter-case predictive monitoring with interplay* among cases. For example, as shown in Figure 3, when two customers submit a form almost at the same time and there is only one Junior Employee who can review the form and another one to input it to the system, the completion time required for the customer who submitted later will be longer compared to the first customer.

However, the dependency among traces could also be exploited without making it explicit in the encoding but rather, learn it from the implicit relations in the training set. We call this as *inter-case without interplay or inter-case for independent cases*. An

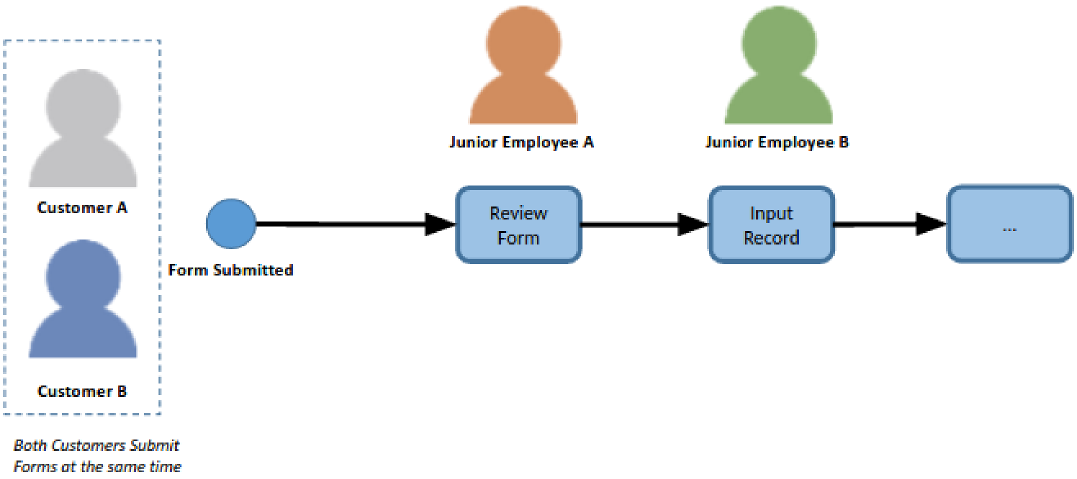


Figure 3. Contention in Resources

example of implied relations are those cases that happen on the same day. We take the frequency of how many cases that occurred in each day to be able to make predictions for the workload on the next day. This is done through the aggregation of the data of each traces in the event log.

Furthermore, cells with "?" values means that based on our research in the state of the art, no current approaches has been done to make the prediction while cells with red text on it are new methods proposed in this thesis. Cells in the blue background have implemented methods in the tool.

This categorization of predictive process monitoring approaches addresses the first research question which is to look into the state of the art and classify the current methods to identify the gap and decide which methods are suited to be included in the Predictive Process Analytics tool.

4.2 General Framework for a Predictive Process Evaluation Tool

To create a general framework for an evaluation tool, we identified five main modules with an addition of a storage for the data sources. This is shown in Figure 4.

First, we have the *Front-end application* which acts as the interface for the user to select settings used for prediction and to analyze the prediction results. The second module is the *Log Manager* which is responsible for managing the logs. Uploading and retrieving the logs are the basic operations of this module. The third module is the *Encoder* which retrieves the log from the storage, parses it and prepares the log for the training phase in the *Predictive* Module. In this part, we retrieve the encoded data, split it into training and test set for evaluation, and build the predictive model from the training data. Finally, using the test data, we test it against the model created to get its accuracy. The aggregation of the results are done in the *Evaluation Module* which is tasked to calculate the error of the prediction model created.

To be programming language agnostic in the encoding and prediction modules, we present the idea of using web services for each algorithm. This explains the multiple clouds for both modules in the framework in Figure 4.

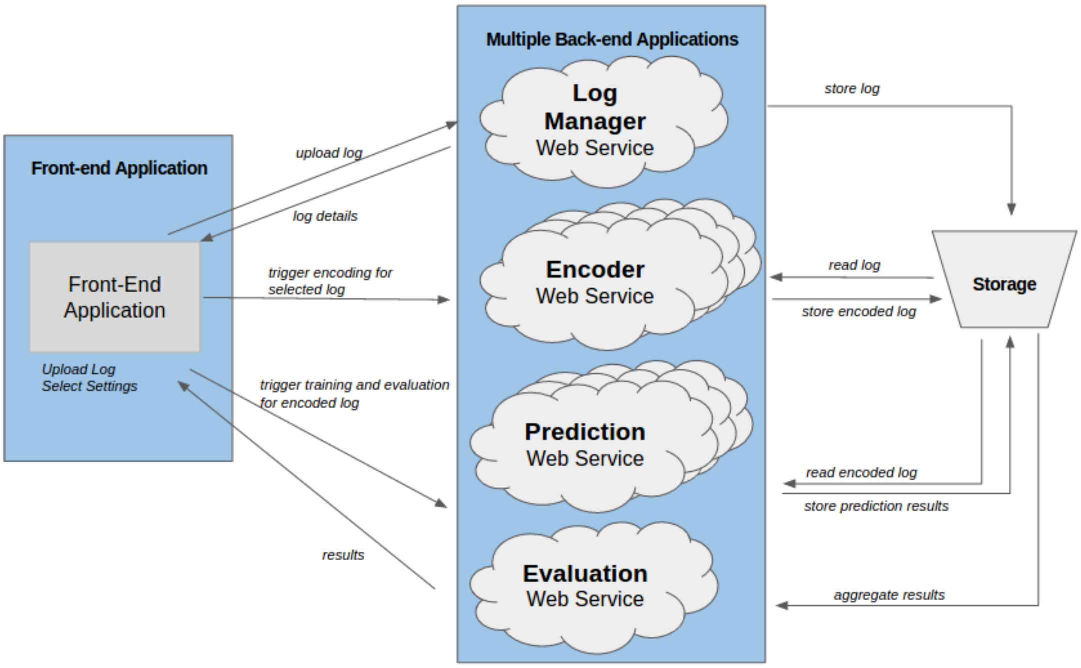


Figure 4. General Framework of a Predictive Process Analytics Tool

4.3 Types of Prediction

Using the cells with blue background in Table 1, we present the following prediction types included in the tool. We also present an architecture on how one can perform a prediction model based on the type of prediction.

4.3.1 Remaining Time Predictions

Remaining time predictions pertain to address the queries on how much longer one has to wait for the process to finish or to reach a certain point in the process. Figure 5 shows the main flow evaluating this metric. During the encoding process of this prediction type, the label is given the value of the computed remaining time per trace. Furthermore, the number of rows in the encoded data for each trace depends on its number of activities. This means that we perform encoding at every event in the trace. After the encoding process, we use this data to be able to create a prediction model. As remaining time is expressed in numerical terms, we use regression algorithms to build the prediction model. Once we have the prediction model, we use the same encoding method for an ongoing trace and feed its output to the predictive model to provide a remaining time prediction.

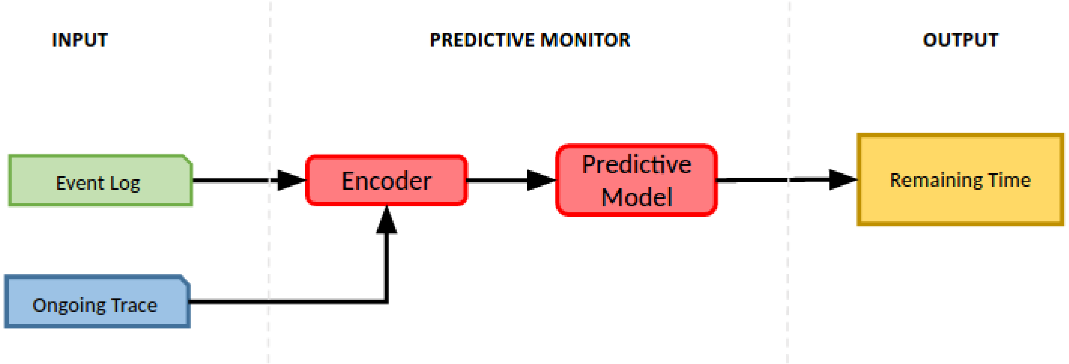


Figure 5. Remaining Time Prediction Architecture

An example use case for this scenario is one in which we want to know when a patient will be able to get the results after performing a blood test examination. We use the historical data of other traces that performed similar tasks for training a predictive model and be able to make a prediction for this instance based on its own features.

To be able to get a prediction of this type, we use two methods called the intra-case prediction and inter-case prediction. The methods differ in the way they encode the event log. In intra-case prediction, it uses the methods presented in the Section 3.1.2 while inter-case prediction uses a novel idea that we present in Section 4.3.5.

4.3.2 Load Prediction

Load Prediction refers to predictions where features are aggregated from multiple traces. An example of this scenario would answer the question of how many cases is expected to be executed tomorrow. Another example for this is to know how many resources would be recommended based on the number of cases expected. The architecture for performing predictions of this type is shown in Figure 6. An event log is fed into our proposed daily calculator encoder where its output is used in a time series forecasting algorithm to build a prediction model. To make the predictions, a user can specify the number of steps of how many intervals to forecast. In this thesis, we only predict one step ahead for every predictive model built. Thus, given a data of the first five days, we predict the load on the sixth day.

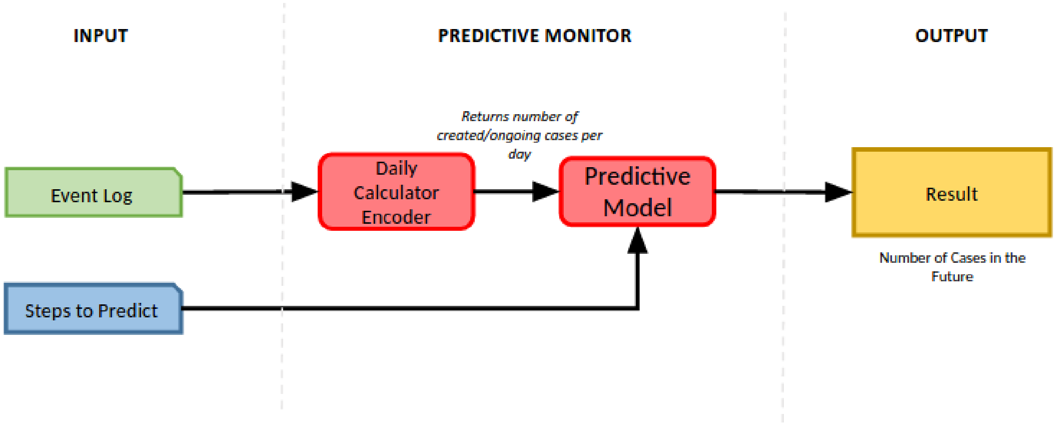


Figure 6. Workload Prediction Architecture

In here, we propose a new encoding method we call as **Load Encoding** which aggregates the data among cases. This encoding method will traverse the log to get all possible dates and/or time and count the frequency of active traces and resources used. Table 2 shows an example of this encoding method. D is the date and time at a given point i to j, where i is the date of the first event execution in the log while j is the detected date of the last event execution. In this thesis, we use the ARMA forecasting algorithm to build the prediction model from the encoded log.

Date and Time	Frequency
D_i	5
...	
D_j	7

Table 2. Load Encoding

4.3.3 Next Activity Prediction

In predicting the next activity, the event log is fed into an encoder. In this thesis, we use the simple index encoding to get the control flow of a given trace for a given prefix. For example, if the prefix provided is three, we expect to have four columns of data in the encoded log. The first three of which are the first three events of the trace and the fourth column is considered as the label or class which is represented by the fourth activity. As we are expecting categorical output in prediction, classification algorithms in machine learning are used with the encoded log.

A use case for this metric is when one wants to know which activity to be performed without looking into the business process model. In general terms, this kind of prediction shows the path to completion of the ongoing process. This may suggest which next activities to perform based on the current state of the running case.

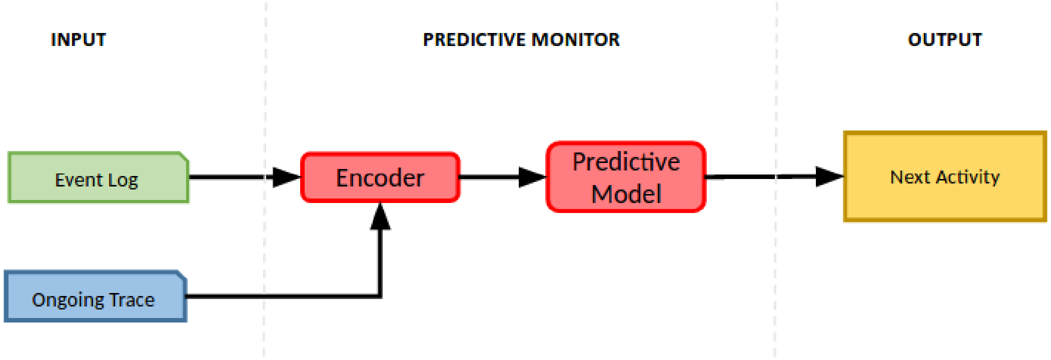


Figure 7. Next Activity Prediction Architecture

4.3.4 Outcome Based Prediction

This type of prediction refers to a categorized prediction results. From one of the papers presented in Section 2.3, a trace can be categorized as fast or slow depending if its remaining time is lesser of higher than a specified threshold time. In this example, we can treat the threshold time as the business goal. A real-life scenario for it is one would like to finish the process in five hours. If the process finishes before five hours, then it is considered as fast, otherwise it is slow.

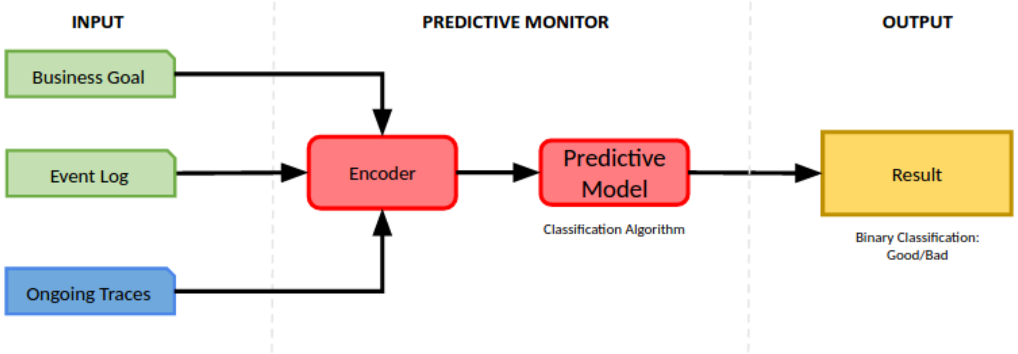


Figure 8. Outcome Prediction Based on Business Goals

Another example use case for this scenario occurs, for instance, when we want to know if one activity be followed by another. For instance, in a hospital setting, we could have a rule according in which surgery would lead to the discharge of the patient. We use the compliance with respect to this business rule to identify for each trace the label class during the encoding of the event log for creating the predictive model.

After encoding the log, we use a classification algorithm in machine learning to train encoded training set. A test trace is then fed to the same encoding algorithm and use the built prediction model to get the predicted classification result. This process is shown in Figure 8.

4.4 Motivation for Inter-case Predictions

When considering predictive process monitoring problems in real-life settings, one can identify several scenarios that reflect different levels of inter-case dependencies. Below, we report four realistic scenarios from an emergency department (ED) and a radiology department (RD). The examples are targeting the remaining time prediction as the prediction type.

1. **Scenario 1: Urgent Patients** An urgent patient who arrives into the ED requires first aid, and thus receives high priority. Hence, the patient does not compete over shared resources, as she gets immediate help. Here, the dependency between the urgent case and all other cases is negligible, and the remaining time of that case will depend on the clinical history of the patient. This is a case where intra-case features are most predictive for the prediction type.

2. **Scenario 2: Homogeneous Patients** Consider the distribution of food among patients in the emergency department. Here, assuming that no patient type has priority over the others, service times are independent and identically distributed for all patients, and the distribution order is random. Hence, we expect that the remaining time for a specific case would depend only on the total number of patients waiting to receive food.

3. **Scenario 3: Heterogeneous Patient Types with Priorities** Consider the radiology department, where patients compete over several types of machines (e.g., XRAY, CT, and MRI). Patients are prioritized according to their diagnosis, age, and recent events. For example, patients who recently went through an Oncology consultation receive the highest priority class. Patients with orthopedic trauma who were recently triaged (seen by a nurse) will receive the lowest priority class. In this setting, it is important to discriminate case types, and consider the number of patients in each priority class [16].

4. **Scenario 4: Heterogeneous Patient Types with History-Dependent Priorities** Consider the surgery department, where patients compete over surgeons and operating rooms. Patients are prioritized according to their diagnosis, age, and their

clinical history. For example, patients with a severe diagnosis and who already had other similar suspected diagnosis or other severe diseases in the past, receive the highest priority class. Patients who never had health problems and with a minor trauma will receive the lowest priority class. In this setting, a more fine-grained case typing than in Scenario 3 must be considered. In particular, one needs to take into account the history of the arriving patients when creating patient priorities.

To summarize, we observe that different processes may require different feature encodings to capture inter-case dependencies. We shall return to these scenarios in Section 4.7, where we present our solution to inter-case feature encoding.

4.5 Bi-dimensional STEP

In this section we present a novel idea on bi-dimensional sequence to feature encoding to answer the question on inter-case dependencies in predictive process monitoring.

4.5.1 Problem Setup

In this section, we present the problem that we solve in this paper, namely the *Sequence-To-feature Encoding Problem* (STEP). The problem arises when we aim at casting the predictive monitoring problem (PPM) into a learning task. The casting appears straightforward. As solution to the PPM, f, one may consider using $\hat{f}$ that results from setting: (i) the training set $\mathcal{S}$ to be L^* (all prefixes of historical cases); (ii) F to be some class of functions (e.g., the set of linear functions); and (iii) r to be some risk function (e.g., the squared error).

However, the training data in the learning task setting, $\mathcal{S}$, is assumed to be a set of independent and identically distributed (i.id.) observations of feature-outcome pairs, (x_i, y_i). In contrast, the training data that stems from L^* contains a set of highly dependent prefix-outcome pairs, $(z_i, y(z_i)), z_i = \phi(\sigma_i, n)$ with $i = 1, \ldots, K, n = 1, \ldots, n_i$: any two prefixes of the same case are highly correlated as they represent the same process execution (intra-case dependencies), and every two prefixes that run in the process at the same time potentially share limited resources (inter-case dependencies). Furthermore, the learning task solution function $\hat{f}$ maps a newly observed feature value to a label, while in the PPM problem, the new value is a (possibly running) case. This leads to the need for transforming sequences into features.

Problem 1. ***(Sequence-To-feature Encoding Problem (STEP))*** *Let L^* be an extended event log that contains all prefixes of the sequences in L. Solving the STEP problem is to find a function $g : \mathcal{E}^* \times \mathcal{Y} \times 2^{\mathcal{L}} \to \mathcal{X} \times \mathcal{Y}$ such that the result of its operation, $\{g(\sigma_i, y(\sigma_i), L^*))\} \subset \mathcal{X} \times \mathcal{Y}$, is an i.id. sample of feature-outcome from $\mathcal{X} \times \mathcal{Y}$.*

Clearly, a STEP solution includes a joint choice of g and $\mathcal{X}$. The STEP has been solved in [19, 12, 11, 22] by various intra-case feature encodings. The main contribution of this paper is a novel solution to STEP that leverages inter-case information, while bounding the feature space size.

4.5.2 Solution

In this section, we show our solution to STEP by means of the construction of a bi-dimensional STEP function g_β that maps every prefix in L^* and its corresponding label into $\mathcal{X} \times \mathcal{Y}$ with $\mathcal{X}$ being a bi-dimensional feature space $\mathcal{X}_1 \times \mathcal{X}_2$. The first component of the feature space, $\mathcal{X}_1$, captures intra-case dependencies, while the second component, $\mathcal{X}_2$, represents inter-case dependencies. The first dimension is defined using existing techniques from [11], while the contribution of this paper is in the definition of $\mathcal{X}_2$, and in the construction of g_β. In the remainder, we specify the basic requirements that a STEP solution needs to satisfy. Then, guided by these requirements, we provide a method to encode the intra- and inter-case dimensions.

Denote $\mathcal{S}^\beta$ the set of feature-outcome pairs such that

$$\mathcal{S}^\beta = \{(x_i, y_i) = g_\beta(\sigma_i, y(\sigma_i), L^*) \mid \sigma_i \in L^*\}. \tag{1}$$

We are looking for a STEP solution such that the following requirements hold:

1. *Sufficiency.* Let σ_i and σ_j be two different prefixes coming from L^*. We require that $(x_i, y_i) = g_\beta(\sigma_i, y(\sigma_i), L^*)$ is independent of $(x_j, y_j) = g_\beta(\sigma_j, y(\sigma_j), L^*)$, i.e., the resulting label y_i depends only on x_i and not on any other (x_j, y_j) in $\mathcal{S}^\beta$.

2. *Accuracy.* When applying the learning task with a class of predefined functions F to the training set that results from g_β, $\mathcal{S}^\beta \subset \mathcal{X} \times \mathcal{Y}$, we require that g_β provides the minimal empirical risk with respect to any other function g that solves the STEP, i.e., for $(x_i, y_i) = g(\sigma_i, y(\sigma_i), L^*)$ we require:

$$(\hat{f}, g_\beta) = \arg\min_{f,g} \frac{1}{K} \sum_{i=1}^{K} r(f(x_i), y_i). \tag{2}$$

3. *Compactness.* We require that the dimension of the feature space $|\mathcal{X}|$ will not exceed the minimal required representation for sufficiency and accuracy to hold. Sufficiency assures that g_β provides sufficient information such that for $(x_i, y_i) \in \mathcal{S}^\beta$, the label y_i will depend only on x_i. This implies that after the encoding, both inter-case and intra-case dependencies required to predict the label y_i are captured by the corresponding x_i. We use sufficiency to justify our decisions when constructing g_β.

Further, accuracy and compactness assure that $\mathcal{X}$ provides an accurate representation of the dependencies, and is compact. In order to demonstrate that the accuracy and compactness requirements hold for our solution g_β, we evaluate the method by using our STEP solution g_β to solve the PPM based on real-world data shown in Section 6. Below, we briefly outline the solution to the intra-case encoding, i.e., the construction of $\mathcal{X}_1$. Then, we focus on the main contribution of our work, namely the inter-case STEP encoding of sequences into $\mathcal{X}_2$.

4.6 Intra-Case STEP Encoding

In this part, we present the construction of $\mathcal{X}_1$. Solving this aspect of the STEP is related to machine learning techniques for predicting labels in sequential data [6]. These methods transform training data that comprises correlated sequences into an independent representation of the same sample by adding relevant information to the observations. In particular, we select the *sliding window method* (Section 4 in [6]) to encode recent (up-to window size w) history of the prefix. We denote by $g_\beta^{(1)}$ the intra-case component of g_β. Then, for $\sigma_i = (e_i^1, \dots, e_i^n)$, we get that

$$g_\beta^{(1)}(\sigma_i, y(\sigma_i), L^*) = ((e_i^{n-w}, \dots, e_i^n), y(\sigma_i)), \tag{3}$$

assuming without loss of generality that $w \leq n$.

Note that one may consider the encoding of additional static and dynamic attribute-values. For example, we set $w = 1$ and choose two AV functions as follows. Let $\alpha_1 : \mathcal{E} \to \mathcal{A}_1$ be the elapsed time from the arrival of the patient until an event (dynamic), and $\alpha_2 : \mathcal{E} \to \mathcal{A}_2$ be the age of a patient at an event (static). Then, an encoding of σ_i above considers both AV functions and the last event (since $w = 1$) is:

$$g_\beta^{(1)}(\sigma_i, y(\sigma_i)) = (e_i^n, \alpha_1(e_i^n), \alpha_2(e_i^n), y(\sigma_i)). \tag{4}$$

The sliding window approach assumes that the last w events (and their Attribute-Values) are sufficient to explain the prediction type. Therefore, if this assumption holds, the sufficiency requirement is satisfied.

4.7 Inter-Case STEP Encoding

We now turn to introduce the inter-case STEP encoding. To give intuition, we use Figure 9, which revisits the four scenarios from Section.

The vertical axis corresponds to the level of dependencies between the predicted label and intra-case features (e.g., recent history, elapsed time, and age), while the horizontal axis captures the level of dependencies for inter-case features (e.g., the number of acute patients in the ED). The blue circle corresponds to the target patient, whose

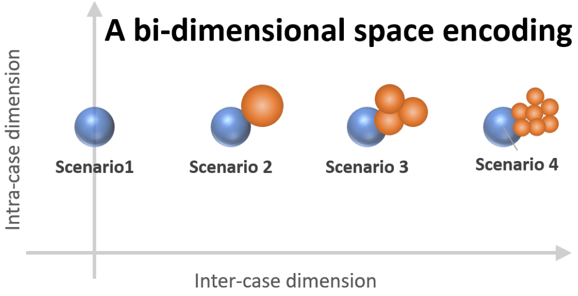


Figure 9. A graphical representation of our proposed bi-dimensional encodings.

prediction type we wish to predict, while the red circles correspond to the patient types on whom the prediction type may depend (*case types*). Each case type can be described in terms of intra-case features and the more case types we have, the more fine grained information is required (i.e., the more features are required for encoding this information). In this setting, Scenario 1 is placed on the intra-case axis, as there is no inter-case dimension. In Scenario 2, the intra-case component does not change with respect to Scenario 1, while the inter-case dimension increases in order to take into account all the other patients grouped in the same class (i.e., all of the same case type). Finally, for Scenarios 3 and 4, the inter-case component is further strengthened by means of a more fine-grained case partitioning into types.

A strong assumption that drives our method is that case types (e.g., urgency priorities) explain the inter-case dependencies between cases. If this assumption holds and the types are properly selected, the sufficiency requirement is satisfied.

Our inter-case STEP encoding relies on four basic concepts, namely *case types*, *discrimination*, *partition*, and (feature) *derivation*. More formally, we consider the encoding of $\sigma = (e^1, \ldots, e^n) \in L^*$ with the timestamp of the most recent event, e^n, being $t_\sigma \in \mathbb{TS}$. We assume that all events are timestamped with an AV function τ. Considering time when encoding inter-case dependencies is crucial, since we assume that these exist only between cases that run in the process at the same time. Below, we go over the four concepts of the solution:

- *Case Types.* Denote $\mathcal{T}$ the set of m case types. Returning to our emergency department, a type of a patient can be set according to her severity grade.

- *Discrimination.* The discrimination function δ is used to set the features that distinguish between case types. In particular, the discrimination function $\delta(\sigma) \in \mathcal{T}$, maps a case σ into its type T. Note that AV functions can be used for discrimination: we may consider, for example, the age of patient, and the elapsed time at the previous event. The typed event log is

$$L_T^* = \{(\sigma, T) \subseteq \mathcal{E}^* \times \mathcal{T} \mid \sigma \in L^* \wedge \delta(\sigma) = T\}. \quad (5)$$

- *Partitioning.* In order to categorize cases into types, we define a function π that partitions a typed event log L_T^* into m event logs according to their types, namely $\pi(L_T^*) \in \mathcal{L}^m$.
- *Derivation.* A derivation function γ maps m event logs for some time t_σ into the desired feature space $\mathcal{X}_2$, i.e., $\gamma : \mathcal{L}^m \times \mathbb{TS} \to \mathcal{X}_2$. For example, γ can produce the feature *number of type* $i = 1, \ldots, m$ *patients in the ED at time* t.

To demonstrate the specification of the four concepts, consider Scenario 2 (i.e., food distribution in the ED), where all patients are assumed homogeneous. Trivially, the case type set is $\mathcal{T} = \{Patient\}$, and the discrimination function results in a typed event log $L_T^* = \{(\sigma, Patient) \mid \sigma \in L^*\}$ as all patients are of the same type ('Patient'). The partitioning function returns L_T^* itself, while the derivation function γ can be set to the number of cases by type at time t:

$$\gamma(L_T^*, t_\sigma) = |\{(\sigma, T) \in L_T^* \mid \tau(e^n) \leq t_\sigma \wedge \delta(\sigma) = T\}|. \tag{6}$$

Here, the inter-case feature space results in $\mathcal{X}_2 = \mathbb{N}$ (i.e., the natural numbers). In essence, once the quadruplet $(\mathcal{T}, \delta, \pi, \gamma)$ is specified, the inter-case component of g_β, which we denote by $g_\beta^{(2)}$, is constructed as follows:

$$g_\beta^{(2)}(\sigma, y(\sigma), L^*) = \gamma(\pi(\{\delta(\sigma) \mid \sigma \in L^*\}), \tau(\sigma)). \tag{7}$$

Let us consider a less trivial scenario, Scenario 3, where patients are typed according to their last event (where the event represents the visited department) into m urgency types, $\mathcal{T} = \{e_1, \ldots, e_m \mid e_i \in \mathcal{E}\}$ with $m = |\mathcal{E}|$ and e_i being the possible events. The discrimination function sets $\delta(\sigma) = e^n$, with e^n being the last event of σ. The partitioning is made according to the typed event log that results from the δ, and the derivation function remains as in (Eq. 6). This leads to m features: the number of patients of all possible urgencies at time t_σ.

Each of the components δ, π and γ plays a different role in our method. The partitioning function π classifies cases according to the notion of similarity imposed by the discrimination function δ. To avoid feature space explosion, the derivation function allows an aggregation of the resulting typed event logs. Having fully defined the bi-dimensional STEP solution, we are now ready to test the accuracy and complexity against real-life event logs. Furthermore, this section addresses our second research question on how we can support inter-case predictions.

5 Tool Implementation

In this section, we discuss the features of the created tool with respect to its front-end and back-end applications. We describe its user interface and how it is usable based on the gathered information for previous projects. The aim of this section is to answer our third research question of designing a web based framework to support predictive process monitoring.

5.1 Front-end Application

This application is called PPA (Predictive Process Analytics) Tool. The front-end application is running using the AngularJS Material library[1]. The front end application is divided into three main sections. These are the dashboard, prediction and log sections. The implementation of the front-end application is publicly available[2]. The application can be accessed with the following link[3].

5.1.1 Front-end Application Architecture

Figure 10 shows the architecture of the front-end application. From the root of the application, we create a module which contains its configurations. Inside the configuration, we can specify the routes which get the corresponding views and controllers. Scope is an object that is used to bind the views (in HTML) and controllers (in Javascript). A module also holds different services which can be used in the controllers. These services hold the links that provide connection to the back-end.

In a developer perspective, the most common changes happen to the boxes with the green background. Views are the HTML files while the controllers are JavaScript files that contain the logic of the operations. In addition, the developer is also asked to edit the routes in order to create the Uniform Resource Locator (URL) link to the page for the corresponding view and controller.

---

[1] https://material.angularjs.org/latest/

[2] https://github.com/kerwinjorbina/ppa-front-end

[3] http://193.40.11.57/

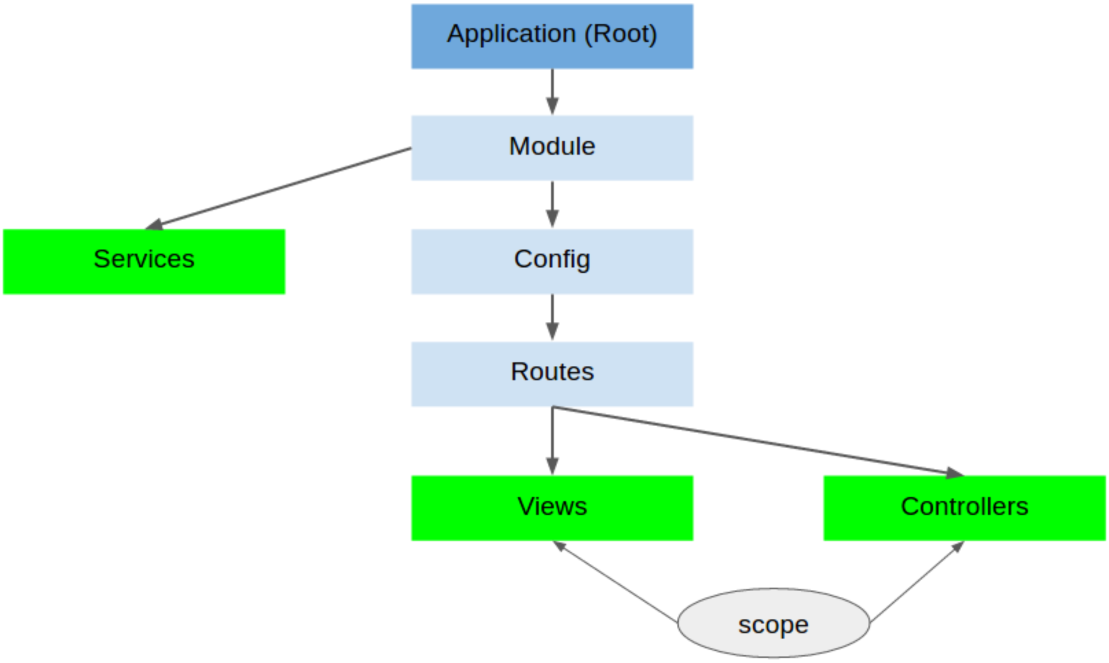


Figure 10. Architecture of the Front-end Application

5.1.2 Views in the Application

The *dashboard page* is the first page the user sees when accessing the application. This shows the details of a log in terms of the number of activities and resources per day and the most executed activities in the log. In this page, a user may select a log to checkout. A screenshot of this page is shown in Figure 11.

The *prediction section* allows the user to select the log, type of prediction and its corresponding method. Refer to Figure 12 for the screenshot of this page. The details of the method are dependent on which type of prediction is selected. The list of the available types of prediction and its coresponding methods are shown in Table 3. The list of logs is provided by a back-end web service which reads the logs in the storage. Once the user provides the selection for the following values, the user will be redirected to the appropriate prediction page. This page is a URL constructor so we can redirect the user to the page of the prediction method.

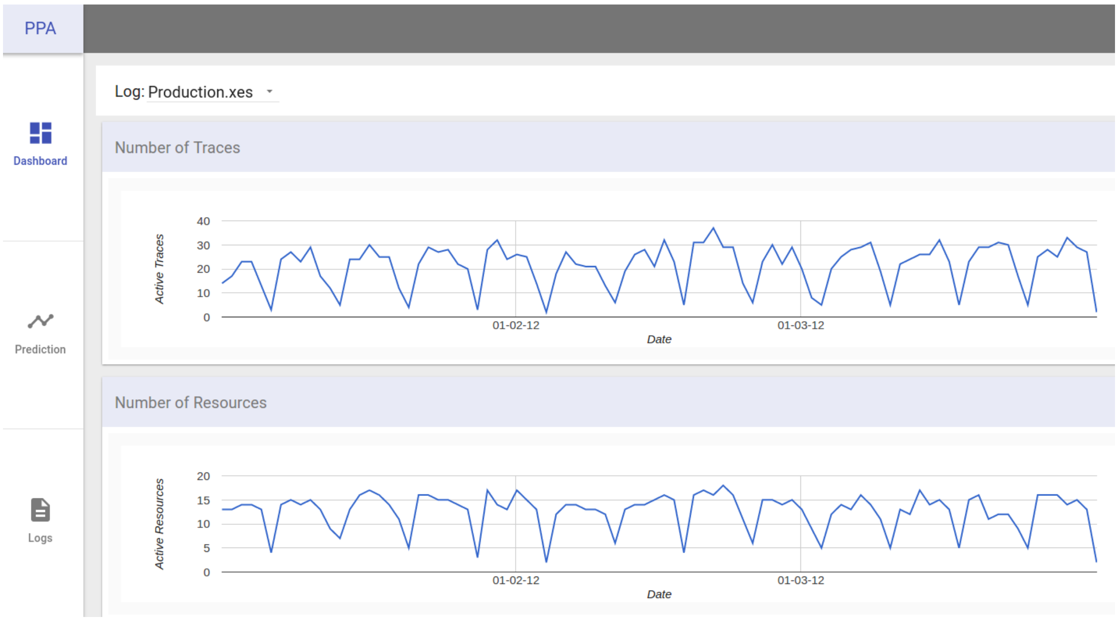


Figure 11. Dashboard Page for the PPA tool

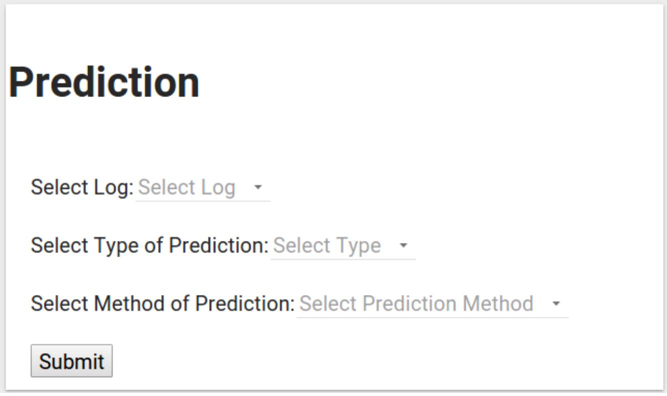
Type of Predictions	Methods
Remaining Time	Intra-case Based Encoding, Inter-case Based Encoding
Load	Time Series Based Prediction
Next Activity	Classification Based Prediction
Outcome	Classification Based Prediction

Table 3. Prediction Types and Methods

The four types of prediction and methods supported (as stated in the contribution section) are the following:

1. **Remaining Time**
    (a) Intra-case Prediction
    (b) Inter-case Prediction

    The two methods for predicting the remaining time share the same HTML file as shown in Figures 13 and 14. This means that both will have the same evaluation parameters. The tool allows the user to look into the general results, evaluation in terms of the Mean Absolute Error (MAE), evaluation in terms of the root Mean Square Error (RMSE) and the predictions per trace versus its actual remaining time value. However, with a specified scope parameter, we are able to distinguish if this page is using inter-case or intra-case encoding.

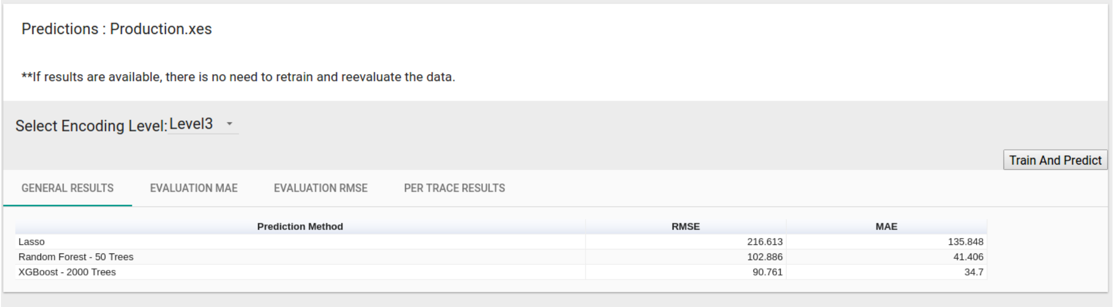

Prediction

Select Log: Select Log

Select Type of Prediction: Select Type

Select Method of Prediction: Select Prediction Method

Figure 12. Prediction Page


Predictions : Production.xes

**If results are available, there is no need to retrain and reevaluate the data.

Select Encoding Level: Level3

Prediction Method	RMSE	MAE
Lasso	216.613	135.848
Random Forest - 50 Trees	102.886	41.406
XGBoost - 2000 Trees	90.761	34.7

Figure 13. Inter-case Prediction Page

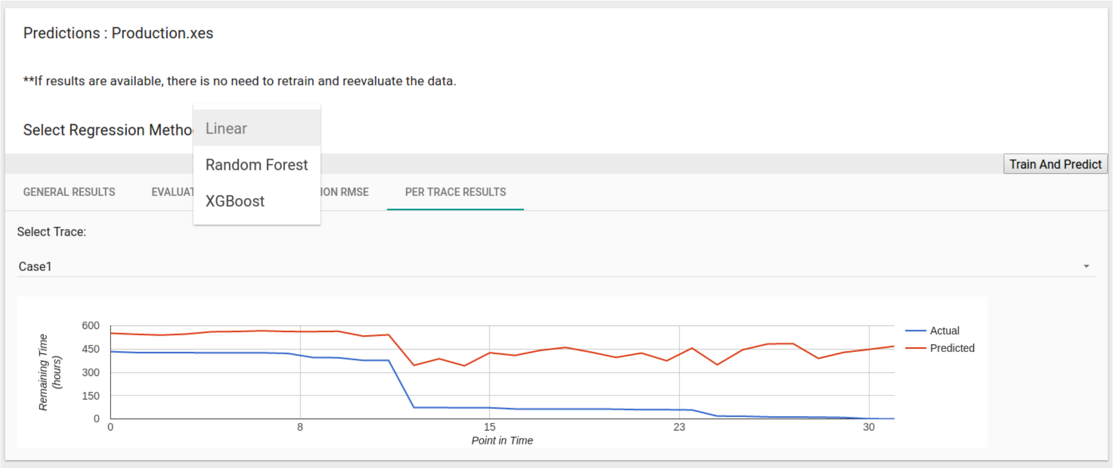
Both methods allow the user to select a setting to visualize the results. In intra-case predictions, we allow the user to select which regression algorithm to use. The regression methods used are Lasso, Random Forest and XGBoost. We use the Scikit library[13] for the first two regression algorithms while the XGBoost library[4] for the third one. While, in the inter-case page, the user may be able to select which of the four encoding levels to use (Level 0, 1, 2, 3). A higher level of encoding means a higher degree of dependency is considered. All the algorithms used are supported in a library implemented in Python.

If no results for a specific log are available on a certain log and method, the user is prompted to click on the Train and Predict Button. Once the user performs this button click, encoding of the log begins (index-based encoding for intra-case predictions and feature to sequence encoding for inter-case prediction). Once encoding is done, it creates the prediction model to be able to make predictions

---

[4] https://github.com/dmlc/xgboost

for the ongoing traces.



Predictions : Production.xes

**If results are available, there is no need to retrain and reevaluate the data.

Select Regression Metho

Select Trace:

Case1

Figure 14. Intra-case Prediction Page

2. **Load**

   Time Series Forecasting

   This prediction type only has one supported method in the tool. Once a submission of this prediction type is done, it performs the load encoding of the log and creates a prediction model for every increment in a day. This means that the current implementation of the tool supports 1-step ahead predictions only. Forecasting for the number of active traces and resources per day is performed. After the training and evaluation are done for the log in this type of prediction, the results are presented in a graph as shown in Figure 15.

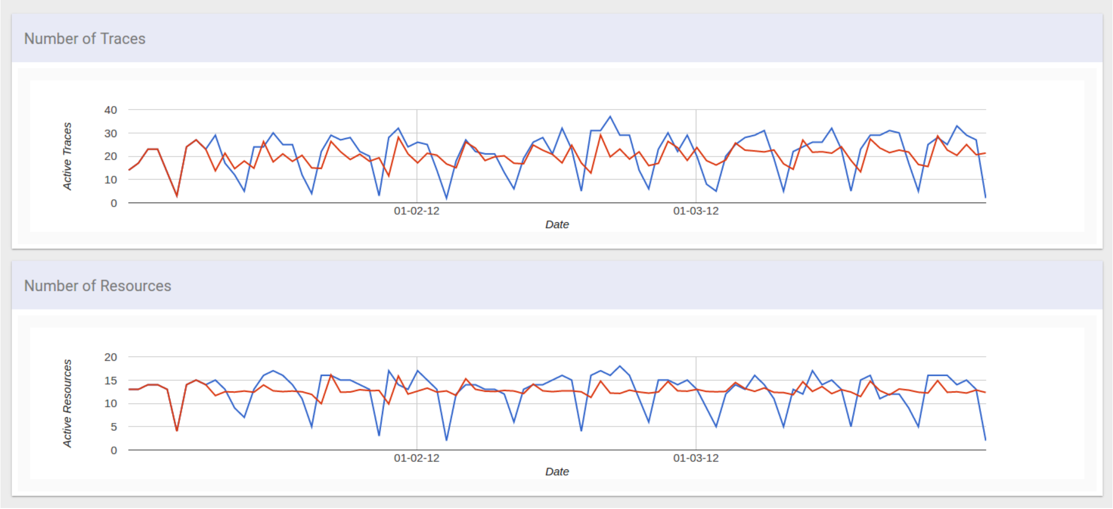


Figure 15. Load Prediction Page

3. **Next Activity**

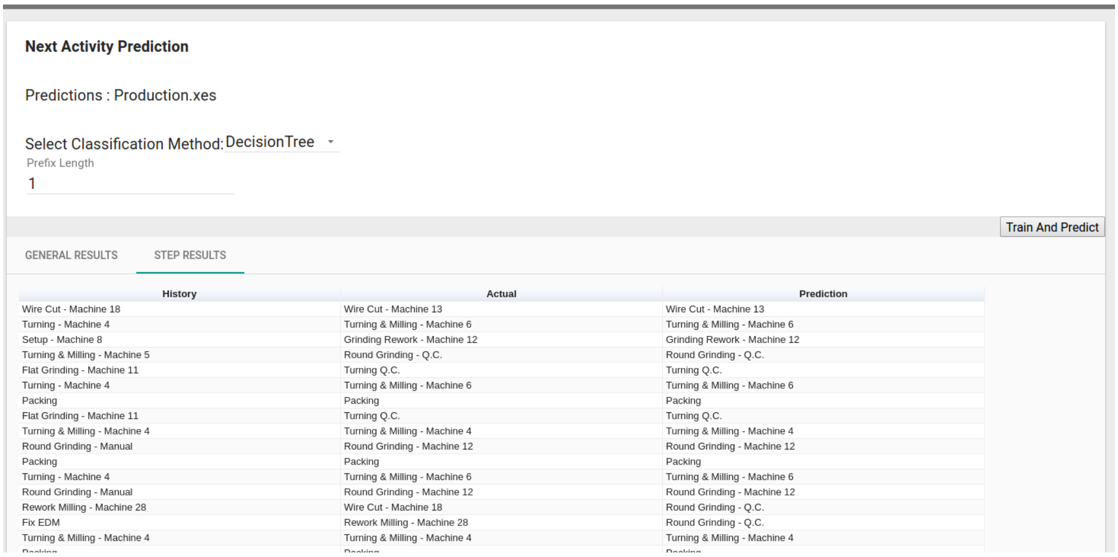
   (a) Classification Prediction

   In this type of prediction, we allow the user to select a prefix, which serves as the cutoff point before the prediction. After the user selects the prefix and selects to Train and Predict the dataset, the results in terms of accuracy are shown in the page and a table of the prediction of each case in the test set. This method uses the Weka Library [9] in Java to create the prediction models. The page for this type of prediction is shown in Figure 16. This page has two tabs, the first is the general results which shows the accuracy of the prediction. The second tab shows the history of events per trace where each activity is separated with an underscore "_" character, the actual next activity and the predicted activity.

4. **Outcome**

   (a) Classification Prediction

   This implementation is a simplified version of the tool created by Federici et. al. [5]. Here, the user is allowed to select a prefix in this prediction type and a Linear Temporal Logic (LTL) rule for the encoding of the log. Once the user selects these parameters and clicks the Train and Predict button, it presents the same visualization the next activity prediction with an addition of a pie chart for the binary classification of the labels. In this project, we only support one LTL rule which is to classify fast or slow traces. To perform this, we use the index



Predictions : Production.xes

Select Classification Method: DecisionTree

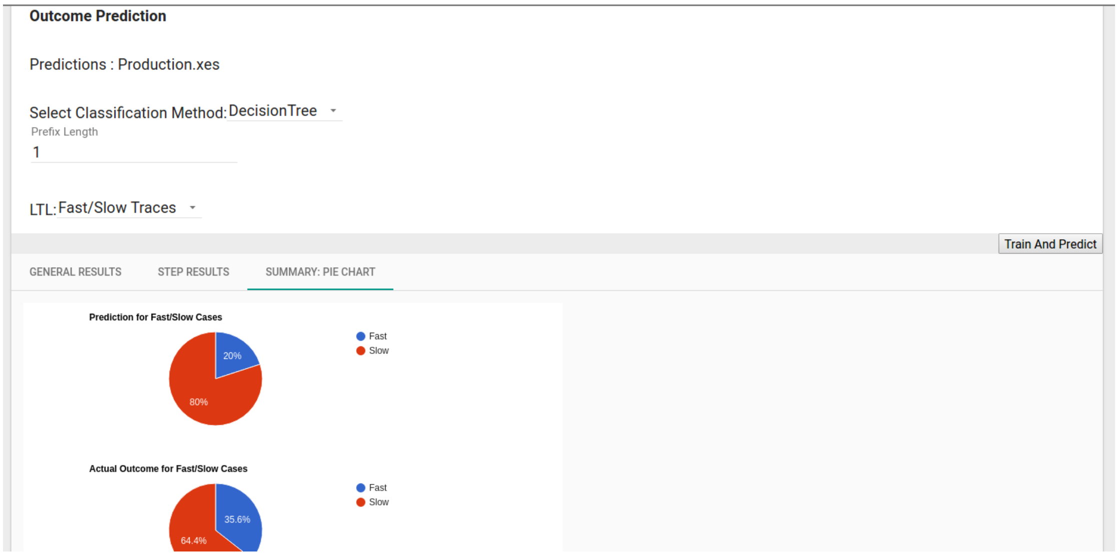
Prefix Length: 1

History	Actual	Prediction
Wire Cut - Machine 18	Wire Cut - Machine 13	Wire Cut - Machine 13
Turning - Machine 4	Turning & Milling - Machine 6	Turning & Milling - Machine 6
Setup - Machine 8	Grinding Rework - Machine 12	Grinding Rework - Machine 12
Turning & Milling - Machine 5	Round Grinding - Q.C.	Round Grinding - Q.C.
Flat Grinding - Machine 11	Turning Q.C.	Turning Q.C.
Turning - Machine 4	Turning & Milling - Machine 6	Turning & Milling - Machine 6
Packing	Packing	Packing
Flat Grinding - Machine 11	Turning Q.C.	Turning Q.C.
Turning & Milling - Machine 4	Turning & Milling - Machine 4	Turning & Milling - Machine 4
Round Grinding - Manual	Round Grinding - Machine 12	Round Grinding - Machine 12
Packing	Packing	Packing
Turning - Machine 4	Turning & Milling - Machine 6	Turning & Milling - Machine 6
Round Grinding - Manual	Round Grinding - Machine 12	Round Grinding - Machine 12
Rework Milling - Machine 28	Wire Cut - Machine 18	Round Grinding - Q.C.
Fix EDM	Rework Milling - Machine 28	Round Grinding - Q.C.
Turning & Milling - Machine 4	Turning & Milling - Machine 4	Turning & Milling - Machine 4

Figure 16. Next Activity Prediction Page

based encoding and get the average time of all traces for a given prefix. This is used as the threshold time in which if the trace has a remaining time value less than it, we categorize it as fast, otherwise, it is slow. The screenshot of this page is shown in Figure 17.

The third main page is the *log page* as shown in Figure 18. This allows a user to upload a log for analysis. In this section, the log uploaded must be in XES format for the other sections to interpret.



Outcome Prediction

Predictions : Production.xes

Select Classification Method: DecisionTree

Prefix Length: 1

LTL: Fast/Slow Traces

Figure 17. Outcome Prediction Page

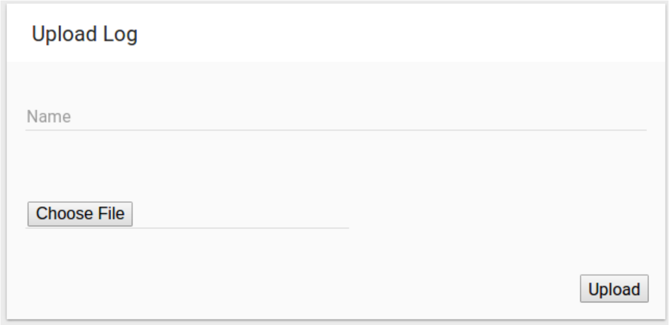


Figure 18. Log Upload Page

5.2 Back-end Application

In this project, we use two backend systems to be able to show that prediction methods can be created separately and can easily be incorporated in the tool.

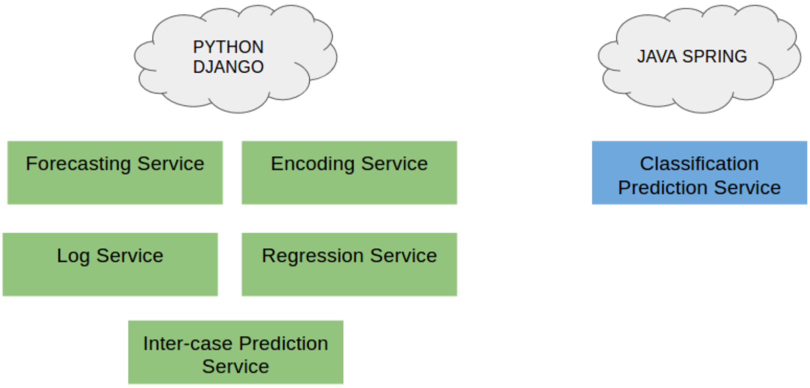


Figure 19. Components of the Back-end Application

The main back-end application is running using the Django Framework[5] in Python. A web service running in Java Spring[6] is used for the web service of the classification algorithms.

For all these prediction types and methods, an automatic separation of the training and testing set is performed in the back-end to the log. For machine learning methods such as regression and classification algorithms, it uses 80 % of the data as the training set and 20 % as the testing set. For forecasting algorithms, it uses the first five data as the initial training set and incrementally increases until the end of the dataset.

5.2.1 Django Back-end

This back-end application presents the following web services:

1. Log Manager - this web service is an interface for the user to upload and get the list of logs. This also provides interfaces to get the load encoding results for the number of active traces and resources for each day. Finally, this also provides the list of most executed events in the log.

2. Encoding - the encoding methods supported by this back-end service are the load encoding, index-based encoding with different labels such as remaining time, next activity and conformance to a business rule.

---

[5] https://www.djangoproject.com/

[6] https://projects.spring.io/spring-framework/

3. Forecasting - the supported forecasting method is ARMA.
4. Regression - the supported prediction methods in this back-end application are the linear, lasso, random forest and xgboost regression algorithms.
5. Inter-case Prediction Service - for the implementation of this prediction method, the sequence to feature encoding, as well as the prediction methods, are integrated into a module in the project.

The implementation of this back-end web service is publicly available[7].

5.2.2 Java Spring Back-end

The java spring application supports the prediction methods for classification. These are the decision trees using the REPTree algorithm, Random Forest Classification and K-Nearest Neighbors (KNN). To be able to run this on a server, we use Docker[8] which is an open source framework that allows an easy deployment by using containers that hold the software. The source code of this implementation is located in this repository[9].

---

[7] https://github.com/kerwinjorbina/predictive-process-webservice

[8] https://www.docker.com/

[9] https://github.com/kerwinjorbina/ppa-spring-backend

6 Evaluation and Verification

In this section, we use real-world event log used to create predictive models and evaluate them. We present the metrics we chose to evaluate the results. In this section, we only present the evaluations of the new implementation performed in this study, *i.e.* the inter-case prediction and the load forecasting.

6.1 Dataset

DS: This is a log pertaining to the manufacturing industry. This is published as a comma separated value dataset that we converted to XES using Disco[10]. This dataset contains the activities performed by machines and human resources to produce a product. The dataset spans from January 2012 to March 2012 with 225 cases, 55 types of activities, 49 human resources and 31 physical resources such as machines.

6.2 Evaluation Measures

As this project is a tool that will help researchers in creating fast evaluations for baseline results in their study, we used the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) to describe the goodness of the prediction models created.

6.2.1 Root Mean Square Error

Root Mean Square Error (RMSE) measures the difference between the predicted and the observed value. This is a general purpose metric for evaluating errors with numerical values. The RMSE is calculated with the formula below

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \hat{x_i})^2}$$

where n is the number of rows in the dataset, x_i is the predicted value at i and $\hat{x_i}$ is the actual value.

6.2.2 Mean Absolute Error

Mean Absolute Error (MAE) measures the closeness between the forecasted and the actual values through the formula:

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|x_i - \hat{x_i}|$$

---

[10] https://fluxicon.com/disco/

6.3 Results

In this section, we present the results of the new prediction methods.

6.3.1 Inter-case Prediction Results

The results for this method are shown in Table 4. The columns Lasso, RF and XGB in the table refers to the methods of prediction. The rows present the different levels of encoding for this method where Levels 1, 2 and 3 refer to encoding method introduced in the encoding explanation in Section 4.3.1. Level 0 presents the results of the base-case scenario in the intra-case dimension while Levels 1, 2 and 3 present the results of the inter-case scenarios. The RMSE and MAE are presented in hours in this production data log. The boldface values present the highest accuracy among all the encoding methods.

Encoding	Measure	Lasso	RF	XGB
Level 0	RMSE	256	195	189
	MAE	136	72	67
Level 1	RMSE	253	158	142
	MAE	137	58	52
Level 2	RMSE	221	121	97
	MAE	137	50	90
Level 3	RMSE	216	109	**91**
	MAE	135	42	**33**

Table 4. Prediction Accuracy on the dataset

From the given results, it can be seen that as the level of encoding increases, the more accurate the prediction results are. This can be explained by the hypothesis that the cases in this log have a strong dependency on each other. In addition, among the prediction methods, XGBoost outperforms the other two regression methods for the encoded log. It is also interesting to see that the non-linear regression algorithms(RF and XGB) present a better result than the Linear (Lasso) algorithm. The reason behind this result can be explained by the non-linear relationship of the remaining time of a running trace and the features for the bi-dimensional encoding. We can say that by taking the inter-case relationship of the traces yield to an increase in prediction accuracy with a 51.9 % increase in the RMSE and 50.7 % in MAE for the XGBoost result between Level 0 and Level 3.

Next, we investigated the influence of the predicted remaining time value on the error. Specifically, we focused our attention to the best machine learning method (XGBoost), running on our chosen dataset. Figure 20 presents the RMSE (in hours) as a

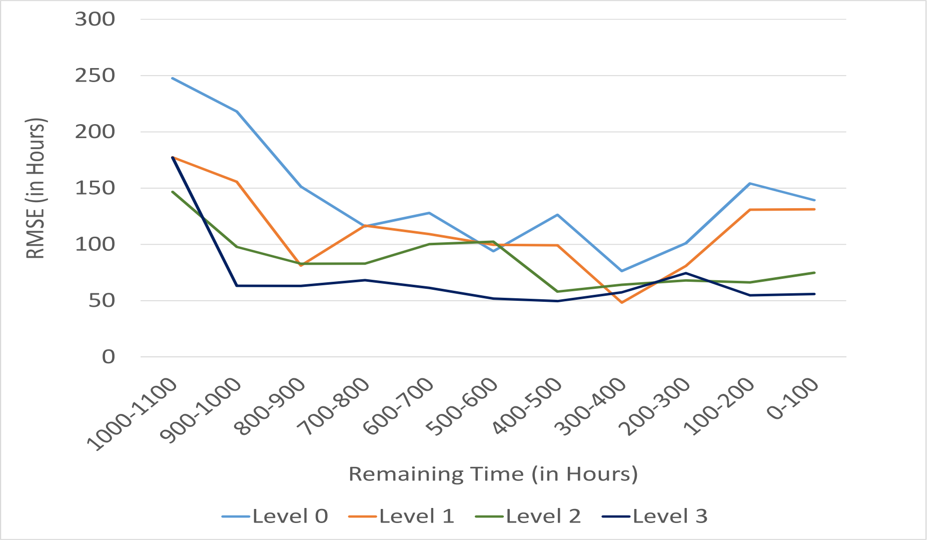


Figure 20. XGBoost RMSE errors for different prediction times

function of the remaining times in the test set. Note that we did not provide the same graph for MAE as they both provide the same trends. Observing the plot, we conclude the following results: (1) as expected, accuracy is worse for longer remaining times (e.g., $> 1\,000$ hours) for all levels; (2) few (or none) inter-case features cause that the prediction accuracy increases as we get more information about the running case, although with some oscillations; (3) more inter-case features guarantee that the accuracy stabilizes faster and remains stable also for short remaining times. For instance, LEVEL3 has almost constantly lower RMSE values compared to the other encodings. This result confirms and strengthens the aggregated results shown in Table 4.

6.3.2 Load Forecasting Results

In this prediction type, we build a forecasting model at every increment of the data. The graphical results of this implementation can be visualized in the screenshot shown in Figure 15. We evaluated two aggregated metrics computed from the log: the number of active traces and the number of resources per day. We analyze these predictions using the same evaluation metrics (RMSE and MAE). The results for the production log dataset are shown in Table 5. The errors are in terms of the frequency count for both metrics.

From the evaluation results and the graph, we can observe that the results of ARMA forecasting are able to predict the load for the next day with a deviation of less than 9 error points for the active traces and 4 for the resources. In addition, it is interesting to see in Figure 15 that the prediction of resources does not exceed the maximum number

Metric	RMSE	MAE
Active Traces Per Day	8.77	7.43
Resources Per Day	3.96	2.91

Table 5. Load Prediction Results with the Production Log Dataset

of resources (18 resources on February 22, 2012). These predictions will provide an interesting information to the production company whether to add or reduce their number of resources on a given day.

7 Conclusion

The main points faced in this thesis are the following:

1. Categorize Predictive Process Monitoring Methods and Types of Predictions
2. Support inter-case prediction feature encoding
3. Design and build a Predictive Process Analytics tool supporting the selected methods

The thesis proposes a tool for supporting in predictive process monitoring. Specifically, the tool supports users in selecting the preferred prediction method from the list of implemented methods, building their own prediction model and getting the predictions. To this end, we looked into the different types of predictions in predictive process monitoring. In the related literature in particular, we looked at two main categories: intra-case and inter-case predictions.

We focused on the inter-case dimension to provide predictions based on the dependencies of cases. We proposed two new encoding methods to be used in predictive process monitoring which are the feature to sequence and load encoding. Moreover, we presented the use of time series in predicting methods for aggregated metrics such as active cases and resource count. Our experiments for the sequence to feature encoding demonstrated improvements in the results of around 50 % with respect to the Intra-case approach when predicting the remaining time in cases.

We were able to see different methods for achieving prediction results. A variety of encoding methods and prediction model creations were shown. From them, we chose specific realms to include in the tool implementation. By looking into the architecture of these methods, we were able to create the architecture of our platform to provide a tool that is able to provide different types of prediction.

In the future,we have multiple directions we would like to investigate. We would like to make improvements in the inter-case approach, specically relating to memory management when building the models. In terms of the tool, we plan to add the capability to provide predictions for prediction types not yet included in the tool and their corresponding methods. Moreover, we would also like to improve the visualization of the results and the presentation of the aggregated results. We also would like to give more options to the user to create more settings (hyperparameters) when building their own prediction models. Overall, we would like to improve the tool to be able to have a runtime prediction interface which can directly use the prediction models built in this tool.

8 Acknowledgement

First of all, I would like to thank my family for the support and encouragement for me to pursue my goal of attaining a Master's degree. Secondly, I would like to thank my supervisors Fabrizio Maria Maggi, Chiara Di Francescomarino and Chiara Ghidini for their patience and teachings. I would also like to extend my gratitude to Arik Senderovich for imparting me his knowledge and helping me in answering the goal of inter-case predictions in this thesis. Furthermore, I would like to give my appreciation to the Fondazione Bruno Kessler - Process and Data Intelligence group in Trento, Italy for hosting me for one semester to perform components of this study.

I would also like to thank professors from the University of Tartu for giving me support in the implementation of the application for this project. Also, to my friends who wished me luck all the way during the entire duration of making this thesis. I would also like to thank my colleagues at Evocon for their understanding and flexibility in allowing me to work with them while doing my thesis.

Finally, I would like to thank the IT Academy of Estonia for giving me financial support for the duration of this Master's programme.

References

[1] Malú Castellanos, Norman Salazar, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Predictive business operations management. *IJCSE*, 2(5/6):292–301, 2006.

[2] Michelangelo Ceci, Pasqua Fabiana Lanotte, Fabio Fumarola, Dario Pietro Cavallo, and Donato Malerba. Completion time and next activity prediction of processes using sequential pattern mining. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8777 of *Lecture Notes in Computer Science*, pages 49–61. Springer, 2014.

[3] Raffaele Conforti, Massimiliano de Leoni, Marcello La Rosa, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. A recommendation system for predicting risks across multiple business process instances. *Decision Support Systems*, 69:1–19, 2015.

[4] Raffaele Conforti, Sven Fink, Jonas Manderscheid, and Maximilian Röglinger. PRISM - A predictive risk monitoring approach for business processes. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, volume 9850 of *Lecture Notes in Computer Science*, pages 383–400. Springer, 2016.

[5] Chiara Di Francescomarino, Marlon Dumas, Marco Federici, Chiara Ghidini, Fabrizio Maria Maggi, and Williams Rizzi. Predictive business process monitoring framework with hyperparameter optimization. In *International Conference on Advanced Information Systems Engineering*, pages 361–376. Springer, 2016.

[6] Thomas G Dietterich. Machine learning for sequential data: A review. In *Joint IAPR International Workshops on SPR and SSPR*, pages 15–30. Springer, 2002.

[7] Marco Federici, Williams Rizzi, Chiara Di Francescomarino, Marlon Dumas, Chiara Ghidini, Fabrizio Maria Maggi, and Irene Teinemaa. A prom operational support provider for predictive monitoring of business processes. In Florian Daniel and Stefan Zugal, editors, *Proceedings of the BPM Demo Session 2015 Co-located with the 13th International Conference on Business Process Management (BPM 2015), Innsbruck, Austria, September 2, 2015.*, volume 1418 of *CEUR Workshop Proceedings*, pages 1–5. CEUR-WS.org, 2015.

[8] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[9] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[11] Anna Leontjeva, Raffaele Conforti, Chiara Di Francescomarino, Marlon Dumas, and Fabrizio Maria Maggi. Complex symbolic sequence encodings for predictive monitoring of business processes. In Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich, editors, *Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings*, volume 9253 of *Lecture Notes in Computer Science*, pages 297–313. Springer, 2015.

[12] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. *Predictive Monitoring of Business Processes*, pages 457–472. Springer International Publishing, Cham, 2014.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. Data-aware remaining time prediction of business process instances. In *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, pages 816–823. IEEE, 2014.

[15] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. Time and activity sequence prediction of business process instances. *CoRR*, abs/1602.07566, 2016.

[16] Arik Senderovich, Matthias Weidlich, Avigdor Gal, and Avishai Mandelbaum. Queue mining for delay prediction in multi-class service processes. *Information Systems*, 53:278–295, 2015.

[17] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[18] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In *Proceedings of the 29th*

International Conference on Advanced Information Systems Engineering, page To appear. Springer, 2017.

[19] Irene Teinemaa, Marlon Dumas, Fabrizio Maria Maggi, and Chiara Di Francescomarino. Predictive business process monitoring with structured and unstructured data. In *International Conference on Business Process Management*, pages 401–417. Springer, 2016.

[20] Wil M. P. van der Aalst, M. H. Schonenberg, and Minseok Song. Time prediction based on process mining. *Inf. Syst.*, 36(2):450–475, 2011.

[21] W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer Berlin Heidelberg, 2016.

[22] Boudewijn F. van Dongen, R. A. Crooy, and Wil M. P. van der Aalst. Cycle time prediction: When will this case finally be finished? In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I*, volume 5331 of *Lecture Notes in Computer Science*, pages 319–336. Springer, 2008.

[23] HMW Verbeek, Joos CAM Buijs, Boudewijn F Van Dongen, and Wil MP Van Der Aalst. Xes, xesame, and prom 6. In *Forum at the Conference on Advanced Information Systems Engineering (CAiSE)*, pages 60–75. Springer, 2010.

[24] Ilya Verenich, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Chiara Di Francescomarino. Complex symbolic sequence clustering and multiple classifiers for predictive process monitoring. In *International Conference on Business Process Management*, pages 218–229. Springer International Publishing, 2015.

[25] Moe T Wynn, Wei Zhe Low, Arthur HM ter Hofstede, and Wiebe Nauta. A framework for cost-aware process management: cost reporting and cost prediction. *Journal of Universal Computer Science*, 20(3):406–430, 2014.

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kerwin Jorbina**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   A Web-Based Tool For Predictive Process Analytics

   supervised by Fabrizio Maria Maggi, Chiara Di Francescomarino and Chiara Ghidini
2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 18.05.2017