UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Stanislav Mõškovski

A web-based framework for the evaluation of predictive process monitoring techniques

Bachelor's Thesis (9 ECTS)

Supervisor: Ilya Verenich, MSc

Tartu 2018

A web-based framework for the evaluation of predictive process monitoring techniques

Abstract:

Business process management (BPM) focuses on optimizations of various activities within the organization, with respect to key performance indicators (KPI). An important task among BPM-related activities is process monitoring which aims to make sure that business processes comply with KPIs. Process monitoring can be performed either offline, using historical data to analyze process execution in the past or online, i.e. analyzing event streams in real-time to identify the problems as soon as they arise. Predictive monitoring is an emerging type of online process monitoring that uses historical data to construct a predictive model using various machine learning methods and then applies this model to a live event stream in order to predict the future performance of ongoing process cases.

Various techniques have been proposed to address typical predictive monitoring problems, such as whether this ongoing case will finish on time or what activity will be executed next in the case. Even though many of these techniques have publicly available software implementations, they typically target one specific predictive monitoring problem. Furthermore, due to variations in evaluation procedures (different data splits, different evaluation metrics reported, etc.), users do not have a readily available way to compare predictive accuracy across multiple techniques. Finally, such solutions are targeting experienced users and also consume a lot of users hardware resources to run the simulations. In this thesis, we have built a web application that allows users with various degrees of expertise in the subject to train, validate and compare models to predict multiple KPIs, using a wide range of predictive monitoring techniques proposed in related work. Moreover, the models can be exported for further use. This application runs all of the computations on the server side, thus eliminating the need for the powerful hardware to construct the models. We compare our solution with existing implementations and highlight clear distinctions and differences.

Keywords:

Business process management, Process Mining, Predictive Monitoring, Machine Learning

CERCS:P170 Computer science, numerical analysis, systems, control

Veebipõhine raamistik äriprotsesside jälgimise tehnikate hindamiseks

Lühikokkuvõte:

Äriprotsesside juhtimine keskendub ettevõtte siseste tegevuste optimeerimisele peamiste tulemuslikkuse näitajate suhtes. Protsesside jälgimine on üks äriprotsesside juhtimise osadest, mille eesmärgiks on teha kindlaks, et peamiste tulemuslikkuse näitajate nõuded oleksid täidetud. Ennustuslik protsesside jälgimine (EPJ) on uus protsesside jälgimise tüüp, mis tuleneb *onlain* protsesside jälgimise tehnikast. EPJ kasutab andmeid, mis kirjeldavad varasemalt toimunud äriprotsesse selleks, et konstrueerida masinõppe meetodite abil ennustatavat mudelit. Treenitud mudelit rakendatakse reaalajas toimuvate protsesside voole, selleks et ennustada protsesside käitumist.

EPJ tüüpilisteks ülesanneteks on ennustada, kas antud äriprotsess lõpeb õigeks ajaks või mitte, või milline on järgmine ülesanne, mida hakatakse protsessi raames täitma. Kuigi nende probleemide lahendamiseks on olemas vabavaralisi tarkvara lahendusi, tavaliselt nad keskenduvad ainult ühele eelmainitud probleemidest. Lisaks, sellised lahendused keskenduvad kogenud kasutajatele ja tarbivad palju riistvara ressursse simulatsioonide jooksutamiseks. Antud töö kirjeldab autori poolt loodud veebirakendust, mis lubab erineva kogemustasemega kasutajatel treenida, valideerida ja võrrelda treenitud mudeleid, ning võimaldab ka mitme tulemuslikkuse näitaja ennustamist, kasutades erinevaid EPJ tehnikaid, millest räägitakse täpsemalt 'seotud töö' peatükis. Rakendus jooksutab kõiki simulatsioone serveris, seega ei pea kasutaja omama võimsat riistvara simulatsioonide jooksutamiseks. Lisaks, autor võrdleb oma poolt loodud rakendust juba olemasolevate rakendustega ning toob esile nendevahelisi erinevusi.

Võtmesõnad:

Äriprotsesside juhtimine, Protsessikaeve, Ennustav jälgimine, Masinõpe

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

Contents

1 Introduction

Business process monitoring plays an important role in business process management. Business process monitoring relies on workflow management systems (WFMS), which support the execution of a business process and record activities performed by process participants as a part of the process. The data from WFMS is serialized into event logs and event streams that contain data about each execution of a business process, i.e. a process instance, or a case.

Offline process monitoring is concerned with studying historical event logs in order to analyze process performance issues and discover their root cause. The discovered insights can be then used to improve future process performance. In contrast, *online* process monitoring is applied to event streams to monitor ongoing cases in order to allow process workers and stakeholders to react to performance issues as soon as they arise. *Predictive* monitoring is a novel type of online process monitoring that uses historical data of case executions to construct a model that can accurately predict the future behavior of similar ongoing cases. Such approach allows workers and stakeholders to react to performance issues before they occur and to prevent them.

Predictive monitoring can be divided into two phases. The first phase is about extracting the information from the log file and constructing the model based on the extracted data. The second phase consists of predicting the future behavior of running cases in an event stream.

In this thesis, the first phase is the main focus. Namely, a web application has been developed using an existing command line-based predictive engine proposed in [30]. This application allows us to deliver predictive monitoring to end users, by leveraging the solution as a service model, which allows any user to run simulations in a web browser, without a need of powerful hardware. Web application fully automates the model training procedure and requires only a minimal input from the users. In order to improve user experience and to simplify the process of model evaluation, the application provides such features as automatic data detection and classification, parallel simulation running, results visualizations and model/result exporting.

The rest of the thesis is structured as follows. Section 2 introduces the important concepts that help to understand the topic, such as business process monitoring in general, what is predictive monitoring and how it is related to machine learning. Moreover, this section discusses similar software solutions developed for predictive monitoring. Section 3 introduces functional and non-functional requirements. Furthermore, this section provides an analysis of the requirements to explain the requirements in more detail. Section 4 dives into the application architecture, describes the tools that were used to build the application as well as provides insight into the implementation details. Section 5 presents an example of a use case for the developed application as well as some insights of the performance of the application. Section 6 and section 7 describe the results of the thesis and introduce the points for improving the application.

2 Background and related work

This section includes information about the background and related work that has been done in order to understand the topic.

2.1 Business process monitoring

Business process management is an area of operations management that aims at discovering, modeling, analyzing and optimizing different activities within the organization [10]. For example, the objective of such optimizations can be reducing the cost, execution times and error rates of various business processes [10]. As seen in figure 1, business process management consists of a number of lifecycle stages.
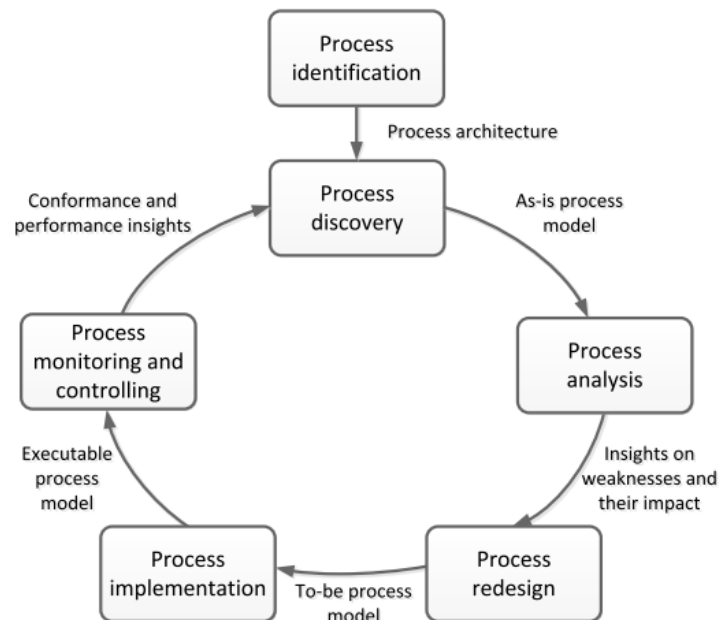

Figure 1. Business process management lifecycle [10].

One of such elements in the business process management lifecycle is the business process monitoring. The core objective of business process monitoring is extracting the information from different data sources that are generated during process execution and verifying that the data conforms to the norms and regulations [10].

In general, data sources for process monitoring come from event logs. The content

of event logs is defined by *extensible event stream (XES)*[1] format, which is the standard format used in process mining. An event log consists of various events that have been performed during the execution of the process, displayed as an entry in the event log. Each entry in an event log represents a state change of the process, such as start or completion of the task [10]. The complete structure of XES can be seen in figure 2.
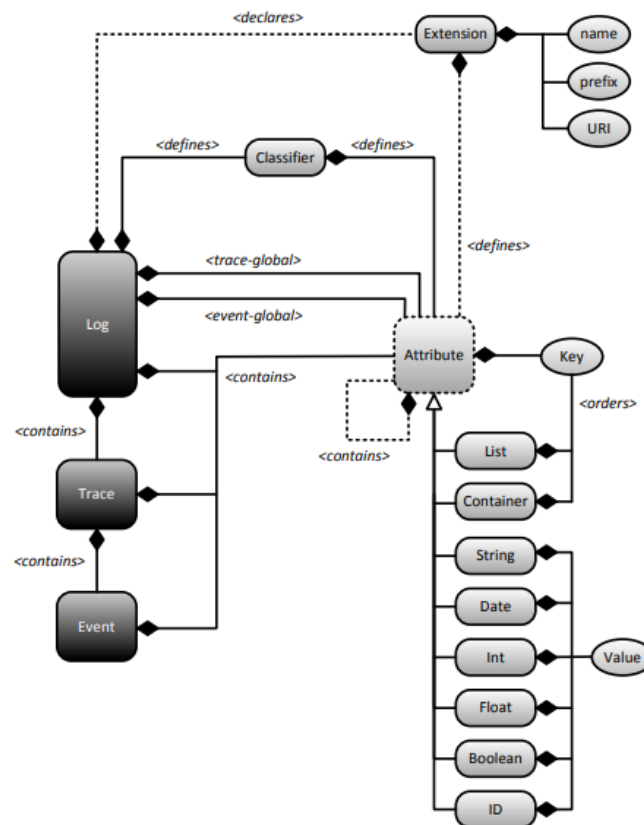

Figure 2. Structure of XES logs [15].

As far as the content entry in event log goes, each entry has the following mandatory fields that are present in every event log: case identifier, activity column and timestamp column.

Case identifier represents a unique identifier for the business process which is used to identify which events belong to a single process in the event log. As mentioned before, case identifier is unique, which means that its relation to events is 1 to *n*. This means

---

[1]http://www.xes-standard.org/_media/xes/xesstandarddefinition-2.0.pdf

7

that there may be multiple events belonging to the same case but a set of events can only have a single parent case.

The activity represents a status of the case. This value may vary for different events in the case, but it is not unique. It is used to represent state change of the case, such as a task start or completion. Alternatively, the column may be named as event name in some event logs.

Another entry that is always present in the event log is the timestamp. As defined by *XES*, a process can have different states, such as the start of the process execution, pausing of the process execution, completion of activity and others. The purpose of the timestamp is to record the point in time when the state change of the process took place. This entry usually includes both date and time and can be used to restore the sequence of events to their chronological order.

Finally, an entry which represents resource is often present but is optional. Resource represents an entity responsible for the execution of the event in a given process. This entity can be a human, a machine or any other object.

It is also important to mention that an event log may contain other entries in addition to those mentioned previously. Those are also optional, just like the resource, but those may be used as features during model construction in order to construct a more accurate description of the process.

Above we have described the structure of the event log defined by *XES*. Event streams are another way to represent the information about the business processes. While event stream is structured the same way as an event log, they represent a potentially unbounded set of data. This is because event logs are stored as files on a filesystem, while event streams act as streams, which are able to consume and produce events in real time.

If event logs contain information about the cases that have already been completed, the event streams contain information about ongoing processes. The former is mainly used for historical analysis of the data such as evaluating bottlenecks in the processes, while the latter is important to provide on-the-fly overview of the performance of the process. With that in mind, process monitoring techniques can be classified into two categories: offline process monitoring and online process monitoring [10].

Offline process monitoring deals with already completed process executions [10]. Event logs are used as an input for offline process monitoring. Event logs contain historical process execution information over a certain period of time, for example, a month, a quarter or a full year [10]. The main objective of offline process monitoring is to identify the reasons for poor performance of the process or to find other bottlenecks that occurred during process execution.

If offline process monitoring focuses on already completed cases, then online process monitoring is concerned with the evaluation of performance of ongoing processes [10]. In this case, a stream of events is usually used as an input, in order to give a real-time overview of the performance of the running cases.

In this thesis, we mainly focus on predictive process monitoring, which is described in the next subsection.

2.2 Predictive monitoring

Online process monitoring, described in the previous subsection, is reactive by nature. It operates on the event streams, consuming the events in real time and detecting issues when they arise. This approach allows managers and stakeholders to solve problems after their occurrence, but it does not allow the users to foresee the problems before they arise.

As mentioned in subsection 2.1, offline process monitoring deals with historical data analysis while the main objective of online process monitoring is the evaluation of the performance ongoing cases. Approaches used in those two process monitoring techniques can be combined, resulting in predictive monitoring.

Predictive monitoring acts as a bridge between online and offline monitoring – it uses data of historical process execution to train machine learning models, which then can be applied to an incoming stream of events to rate the performance of incoming cases. The position of predictive process monitoring, relative to offline and online monitoring, can be seen in figure 3. While online business process monitoring is reactive, predictive process monitoring is proactive. This means that predictive process monitoring is able to detect the problems before they arise, thus giving users an ability to react to an event before it happens, allowing them to either prevent the issue or minimize the losses caused by it.
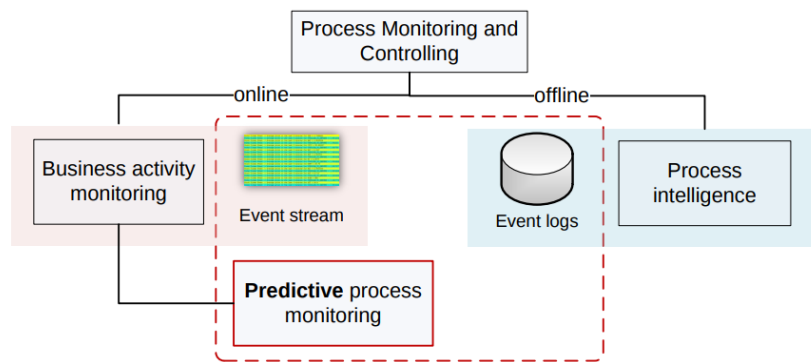

Figure 3. Process monitoring and controlling techniques [29].

Not only does such approach allow one to create machine learning models using different machine learning algorithms, but it also allows construction of specific models that are focused on predicting an attribute that was selected during training. Furthermore, large amounts of the training data with a combination of thoroughly selected machine

learning algorithms can result in a trained model that produces accurate results once applied to a real-time stream.

To sum up, our application facilitates predictive process management, by analyzing historical data provided by the user, in order to train models by using different machine learning algorithms. Such models can then be used for predictions on a real-time stream of events by already existing applications, such as one described in [27].

2.3   Machine learning

Machine learning is a field of science, that draws together different fields of science such as artificial intelligence, probability and statistics, information theory and many others [23]. Machine learning system is described by the training data and by the learning algorithm that extracts useful patterns from the data. In today's world, data is being collected as a result of various activities. Such data can be used as a direct or indirect source of training experience for a machine learning model [23].

In this thesis, the source of the training data is an event log file, described in section 2.1. The learning algorithm extracts information from the event log, generally as feature vectors. From such vectors, a decision tree is constructed, which is used to make predictions.

Decision trees are hierarchical structures, which classify instances by moving from the root node of the decision to the leaf of the tree [23]. Every node represents an attribute of the instance, while every edge moving from the node represents one of the possible values that this attribute can have [23]. So in every node, the attribute of the particular instance is tested and an appropriate edge is chosen. This process is recursive and repeats itself until a leaf node of the subtree, chosen in the previous step, is reached. When a leaf node is reached, this means that an appropriate class has been found for the particular instance.

Random forest is a learner that can be used for both classification and regression and is an ensemble of randomly trained decision trees. When the random forest is used, predictions are made by aggregating the predictions of all of the decision trees in the forest [21]. This means that majority vote method is used for classification, while for regression an average of all votes is used.

Gradient tree boosting is another machine learning technique which is based on decision trees. It is used to solve supervised learning problems and is based on the idea of functional gradient ascent [9]. The main idea behind gradient tree boosting (as it is for all boosting techniques [32]) is boosting weak learners (e.g. decision tree, neural network) into strong ones. In [32], weak learners are defined as component learners that are slightly better than random predictions, while strong learners are able to do very accurate predictions.

Extreme gradient boosting (XGBoost) is a machine learning technique based on gradient tree boosting. One of the core advantages of XGBoost over regular gradient

boosting is scalability: XGBoost runs more than ten times faster on a single machine [7]. This is mainly due to innovations introduced with XGBoost. Those innovations include: handling *sparse data* (cases where most of the row columns are empty), using parallel computing and out-of-core data processing [7]. Furthermore, XGBoost uses model regularization which allows it to avoid over-fitting, giving it better performance [7].

2.4   Similar tools

In order to illustrate what differentiates our solution from already existing tools, let us compare already existing products with the application implemented in this thesis.

Firstly, let us discuss solution implemented in [11]. This application represents a client-side solution that uses *XES* files as an input. One of the core differences of the application described in [11] and the application described in this thesis is that [11] is executed completely on the client-side, while we propose a solution as a service (SaaS). This means that our application is accessible with any modern web browser and the client does not have to have to install any additional dependencies or have a powerful machine in order to receive the output of the application. Moreover, in [11] neural networks are trained to output prediction result, while our solution uses mostly trees-based approach.

Secondly, an extension for *ProM* has been developed by Federici et al. [12]. This plugin gives the user an ability to select both remaining time for the case completion and case outcome (whether or not the case will meet the required deadline). In this application, two of the following encoding methods are provided: *frequency* based and *sequence based*. Furthermore, Weka random forest implementation is used as a learner algorithm. In our proposed application, a more fine-grained choice of training parameters and prediction targets are provided. By default, in [12] user has to provide a configuration for every training process, which might cause issues for users that are inexperienced in machine learning, while in we propose a set of default parameters that can be used by default. On a final note, this application is a hybrid solution, which means that predictions are made on the web server, while the user still is required to use a desktop application, where one has to provide the IP address of the web server as well as port where the predictive engine is running, while we propose a solution that is accessible via web browser.

In addition to the plugin developed in [12], another extension for the *ProM* framework has been developed by de Leoni et al. [8]. This plugin extends the functionality of already existing framework, by providing alternative prediction targets such as next activity in the trace, workload per resource, total workload and others. The plugin uses only decision trees. As mentioned before, this is an extension of a framework, not an end to end solution that we propose in this thesis.

Finally, let us mention an application developed by Kerwin Jorbina in [20]. As indicated before, all of the implemented solutions are either client-side solutions, where the user must use own CPU or GPU resources in order to produce an output, or hybrid

solutions where the user is required to download a third-party desktop application in order to use the functionality. However, this solution is implemented as a server-side application, so no third-party applications are required to use this service. The main difference between application developed in [20] and our proposed solution is that our application uses more mature predictive monitoring technology, that gives the user much more control over the training parameters. The model output in [20] is isolated and incompatible with third-party components, while our framework provides an integration with one of the popular process mining platforms, Apromore.[2] One of the reasons of why this integration is important is that the models that were trained in the training application could be deployed into the runtime component developed in [27], which would allow the models to make predictions in real time on using an incoming stream of events as an input. Not only are the technologies incompatible, but the trained models that are produced as an output of the application are also incompatible with the runtime component that was developed in [27]. In order to fix this, we developed an application that is using technologies that are fully compatible with Apromore and made sure that the models produced in our application were completely compatible with runtime component developed in [27].

---

[2]http://apromore.org/

3 Requirements analysis

The following section of this paper moves on to describe in greater detail the functional and non-functional requirements which were implemented during the application development. In addition, a reasoning is provided on why such functional and non-functional requirements were chosen.

A functional requirement is a requirement to the project which describes a function that a system or component must be able to perform [1]. In order to provide a clear overview of functional requirements, we have been grouped into independent sections:

- **Log uploading**

    1. The system must accept logs in a *comma-separated values (CSV)* format

- **Automatic data attribute categorization**

    1. The system must automatically categorize the data attributes in each column of the event log.

    2. The system must give user a possibility change the type of each column manually.

- **Training parameter selection**

    1. The system must give the user a possibility to regenerate dataset configuration file.

    2. The system must provide a possibility for the user to switch into 'advanced mode'.

- **Prediction types**

    1. The system must allow remaining time prediction.

    2. The system must support next activity prediction.

    3. The system must allow prediction of case outcome (fast/slow case).

- **Advanced mode requirements**

    1. The system must allow manual selection of various sequence encoding and sequence bucketing algorithms as summarized in [30].

    2. The system must allow the user to manually select learners that will be used for predictions.

    3. The system must provide tooltips which user can read to acquire additional information about given algorithm.

- **Results visualization**

  1. The system must provide a visualization of accuracy achieved at different stages of a lifetime of a case.

     (a) The system must allow the user to select an evaluation metric.
     (b) The system must give the user a possibility to compare relevant models (relevant models - trained with the same log and predicted the same outcome).

  2. The system must provide a visualization of feature importance for a given model configuration.

- **Interoperability**

  1. The system must be able to output trained models in a common format (e.g. as a serialized Pickle object) so that they can potentially be reused by other application, for example, to make predictions at runtime using these models.

Non-functional requirement definition, however, is very broad compared to the functional requirement. In this project, we consider a non-functional requirement an improvement to the user experience.

Below one can see the list of non-functional requirements:

1. The system should provide reasonable default parameters for starting the simulation.

2. The system should provide the user with a way to track simulation progress.

3. The system should be responsive – respond in less than 10 seconds and if an action requires more time, the user should be notified about it.

3.1   Log file analysis

Before proceeding to examine the main functionality of the application, it is important to mention log file uploading. Although *XES* file format has been designed in order to interchange event data between various information systems [2], it was decided the application described in this project would use *CSV* as an input file format. The reasoning behind such choice is the tools used in the predictive engine are designed to operate using *CSV* file format as well as tools offering *XES* to *CSV* file conversion being widely available, one of them being Fluxicon Disco[3]. This is due to event logs coming in a variety of different formats such as *csv*, *xes*, *mxml*, and others. From the aforementioned

---

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

file formats, *CSV* is the most common one and there are various converters available that allow conversion between different formats. Moreover, we aimed for a universal solution that is not locked into a specific file type.

3.2 Automatic data attribute classification

Moving on, the second step after log uploading is the automatic categorization of data attributes in each column. In order for a machine learning algorithm to accept the learning data, it has to be classified beforehand. To improve the user experience when using this application, the following process is automated, but the user is allowed to override the categories proposed by the system.

Every entry in the log file represents an event that took place during a case that is present in the log. Knowing this, we can define the minimum required amount of information as following columns: a case identifier (a unique identifier that is same for every event in the same case), activity (an action that has been performed during an event) and a timestamp (when the action has taken place).

As far as other remaining attributes are concerned, they can be categorized into the case and event attributes, which, in turn, can be also categorized into categorical and numerical values.

An attribute is considered to be case attribute when a value of this attribute is the same for all events that belong to a given case. If an attribute has multiple values across different events in a single case, this means that the attribute is considered to be an event attribute.

Next, a case or an event attribute can be classified as a numerical or a categorical attribute. Numerical and categorical attributes can be also named as quantitative and qualitative attributes of an event or case. Quantitative attributes are represented by integers or continuous numbers [28]. Categorical or qualitative attributes are just different nominal values that are used to distinguish one object from another [28]. Such values can be also represented as numbers and in such cases, they should be treated as symbols [28].

3.3 Training modes and prediction types

When gathering the requirements for the application, a lot of emphases was put on the fact that the application should be as simple as possible, while satisfying the needs of both basic and advanced users. With that in mind, in addition to the basic mode, which offers preselected algorithms for the user, an advanced mode has been developed which offers the user a wide variety of machine learning algorithms. In this mode, the user can manually select encoding, bucketing and learner for a single model as well as hyperparameters that will be used when training the model (e.g. maximum depth of a tree when using gradient boosting as a learner). Moreover, this model allows the user to train multiple models in parallel, each with a different configuration.

Based on the *XES* format definitions, following prediction outcomes can be evaluated for every event log: case remaining time, next activity and case outcome (fast/slow case). Those problems are generic and do not require any additional entries and can be evaluated based on the data that is contained in the event log by default.

Remaining time a regression problem. For this prediction target, remaining time until case completion is calculated based on the information about already finished events.

Next activity is a classification problem. The objective of this prediction is to output the next event that will take place in a given case.

Case outcome is a classification problem. This prediction evaluates whether or not the case will meet the expected deadline (is it a fast or a slow case).

All of the aforementioned predictions are applied to all of the cases present in the event log. In addition to those values, event log specific values can also be used as prediction targets due to the nature of machine learning. This is because machine learning algorithms only require data to be in a specific format, in order to be able to search for patterns in the data, and is not interested in the contents of the data.

3.4   Result visualization and model deployment

Result visualization plays an important role in the application. It is essential that result representation the user is informative as well as eye pleasing. Since the output produced by the predictive engine is in form of *CSV* files, it is crucial to transform the output into more easier to understand format. To do this, in this project validation results should be represented as charts. As there are various ways to measure the accuracy of the model, below the ways used in this application are presented.

Firstly, visualization of model accuracy prediction accuracy must be provided. Based on this visualization user can make a decision whether the model is predicting accurately or the model is overfitting the data. Since this metric is different for classification and regression methods, then the visualization methods for those should differ.

Secondly, one should see how the predictive power of a model changes across case lifetime, i.e. as more events get exectuted and more data become available. By using different evaluation metrics, such as mean absolute error, one can see how close the predictions are to the correct values. In this section, it is also important to provide a possibility for a user to compare the models so that one can see which of the trained models makes more accurate predictions.

Finally, the user should be provided with a visualization of how important features were during the prediction. The purpose of this is to give the user an idea how much of an impact a given feature had on prediction, so in the next simulation one could, for instance, exclude a feature from predictions or, on the contrary, include some of the excluded features into predictions.

3.5 Interoperability

In addition to validation results in form of *CSV* files, the predictive engine also produces a model based on the simulation parameters. Both of those outputs are in open data format and thus can be analyzed by external third-party applications. This allows users, for instance, to export the trained model and used in any other application that is able to process *pickle (pkl)* files. Moreover, models exported from the application can be used for predictions on event streams. One of such examples would be the deployment of the trained model into the Nirdizati Runtime component, which is described in [27].

Not only can user export the *pickle* file of the trained model, but the user is also able to export the result files that are generated for the model accuracy report. This allows the user to create own visualizations if for some reason ones provided by our application are not suitable.

In other words, all of the output produced by the application is in open data format and can be consumed by other developers, applications or users without any specific tools provided by our application.

4 Approach and implementation

Purpose of this section is to provide a high-level overview of the architecture of the solution and to give an insight into technologies that were used to complete the project. The solution architecture was designed to support multitenancy. Such approach allows every user to have an own instance of the apllication. Furthermore, instances of the application can be configured individually to suit the needs of different users.

4.1 Architecture

In order to conceive the solution architecture, let us illustrate the flow of the data through the application. Visualization of the data flow can be seen in figure 4.
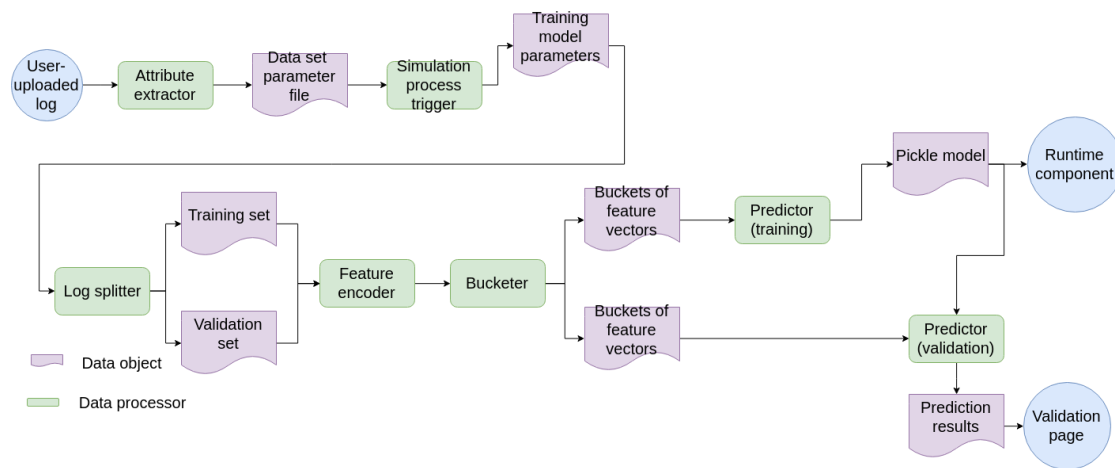
Figure 4. High-level overview of the data flow in the system (adapted from [31])

Before starting operations on the log, it is crucial to extract the required data from it. This is done by the front-end application, as it may require user input in order to verify that the data classification and detection have been done correctly.

Firstly, it is important to verify that the required columns, defined in subsection 3.2, are present in the log file. This inspection is done automatically in order to save users time as well as improve the user experience. If at any point user feels that the values chosen by the system do not correspond to his expectations, one can override the values manually.

After the presence of the minimum required set of data has been verified, attributes of each case are automatically extracted from the log and categorized into case or event attributes and then into numerical and categorical values correspondingly. Just as the previous process, this procedure is fully automated, because if done manually, this

process can be time-consuming. Manual overrides are also allowed here because the user might want to exclude some attributes during predictions to improve prediction accuracy or, in some cases, the attribute might contain 'future values' – values that are currently not known, but might be evaluated during prediction process.

Based on the data gathered in previous procedures, a data set descriptor is generated, which will be used during the prediction process. The descriptor contains the main information about the attributes of the log: names of the required attributes and categories of the remaining attributes. Descriptor also contains information about the columns that will be ignored during prediction as well as 'future values'.

After the data set descriptor has been generated, a simulation process can be triggered by the user in the front-end application. When the process has been triggered, the front-end application generates a simulation process descriptor for the predictive engine. Simulation process descriptor contains information about the hyperparameters for the learning algorithms. The descriptor may also contain the information required by other training parameters, such as a number of clusters to bucket the data into. For example, if a simulation process is triggered with a decision tree as a learner, then the number of maximum features and the maximum depth of the tree are passed into the descriptor.

During the next steps, actions are performed by the prediction engine. Internally, the log is split into the training and validation sets (80% for training and 20% for validation). After the log has been split into the training and validation sets, the features are encoded using the method which was provided by the training descriptor and then, if the bucketing method was provided, collected into buckets. Next, the data is passed to the predictor, which learns on the data and evaluates the results. As an output, the predictive engine generates a model in the *pickle* format and the accuracy report of the model. Both model and the reports are accessible via the front-end of the application. Furthermore, report results can be visualized in the front-end or exported for further analysis outside the application. The export feature is also available for the model, trained by the engine.

4.2 Technologies used

In this section, an overview of different technologies that were used to develop this application. End-product can be divided into two parts: presentation layer written in Kotlin programming language and predictive engine in a form of Python scripts. Furthermore, ZK framework was used to develop user interface of the application.

4.2.1 ZK

ZK is enterprise AJAX framework designed to build desktop-like applications for the web. ZK is owned and developed by Potix, a Taiwanese company which has been founded in 1998 [22].

Since the project is open-source and licensed under LGPL 3.0[4] it was eligible for free ZK EE license, which allowed us to use some of the components that are not available in the free version. ZK EE version 8 was used for the development of the application.

One of the core advantages of ZK is that it is server-centric framework, thus no business logic is ever exposed to the client. Moreover, all of the requests are handled by the framework, which means that user interface can be written in pure Java. This might appear as a limitation, but in fact, ZK also supports client-side programming if needed. The framework uses component-based event-driven architecture to handle the state of the application [26]. This allows developers to create content-rich single page applications.

Component-based means that all of the elements of the application can be divided into independent modules, which communicate with each other using provided interfaces. The benefit of this approach is that it allows containing complexity inside of independent components, allows parallel development of different components as well as improves quality of the product [5].

On the other hand, event-driven means that events are used to change the state of the application. An example of this of this would be a user interaction with the application. On user input, a component emits an event, which can be consumed by other components that are subscribed to the events emitted by a given component.

Another feature provided by ZK is an XML-formatted language ZUL which is an extension to the XUL language [25]. ZUL allows creating different user interface components declaratively without writing any Java code. Figure 5 illustrates declarative user interface composition using ZUL. Not only does it allow to create components, but ZUL also allows attaching listeners, styles and other various component attributes.

In this thesis, both layout generation approaches were used: declaration with ZUL and programmatic layout generation.

```
<dropupload id="dropArea" sclass="drop-area" anchor="${fileNameCont}" detection="browser" onUpload="chooseFile"/>
<vlayout vflex="3">
    <hlayout hflex="min" vflex="min">
        <fileupload sclass="n-upload-btn" id="chooseFile" upload="true" label="${labels.upload.choose.file}"/>
    </hlayout>
    <vlayout style="margin-top: 40px">
        <html sclass="upload-label param-label display-block zk-font" width="30%">${labels.upload.log_support}</html>
    </vlayout>
    <vlayout hflex="min" vflex="min" style="margin-top: 40px">
        <button id="upload" sclass="n-upload-btn" disabled="true" label="${labels.upload.train}"/>
    </vlayout>
</vlayout>
```

Figure 5. Layout declaration using ZUL.

4.2.2 Kotlin

The main language for the user interface development of the project was chosen to be Kotlin. Although it was mentioned that ZK is a Java framework, Kotlin is 100%

---

[4]https://www.gnu.org/licenses/lgpl-3.0.en.html

interoperable with Java and is compiled to the Java virtual machine bytecode [19]. This allows developers to use rich collection of Java libraries and frameworks with Kotlin without any compatibility issues. Furthermore, Kotlin and Java can be used in the same project, thus allowing you to call Kotlin from Java code and vice versa.

Kotlin is an open-source statically typed programming language that targets different platforms one of which is JVM (Java virtual machine) [19]. Kotlin is developed by the company named JetBrains with headquarters in Prague, Czech Republic [16]. First stable release for the language was launched in February of 2016 [4]. This indicates that Kotlin is a rather young language but nonetheless is it developed very actively and currently it is at version 1.2.30. The aforementioned version was used to develop the application described in this thesis.

One of the traits of this language is that it combines both object-oriented and functional paradigms. On one hand, Kotlin has extensive support for object-oriented programming by not only using regular classes like Java but, by going a step further, it introduces data classes, sealed classes and objects. For instance, sealed classes can be used to represent strict class hierarchies and act as an extension to enum classes in Java [18]. On the other hand, higher-order functions is a crucial concept introduced by functional programming and Kotlin provides comprehensive support for it. In Kotlin, one can pass higher-order functions or blocks of code as function parameters which in turn allows the language to have such concepts as function composition and coroutines[5].

The second advantage of Kotlin is that it avoids the problem of nullability by providing a strong type system. In Java, a null value is used to represent the absence of value [14]. However, due to Kotlin's strong type system, references that can contain null have to explicitly be marked as nullable [17]. Such type system minimizes the number of null references in the code because in order to access nullable type one should use *!!* operator in order to assert that a value is non-null or use safe function calls using *?* operator.

Finally, Kotlin is more concise by cutting the number of lines of codes by approximately 40% compared to Java [19]. In this case, having less code means that the code is easier to read and debug. This is achieved, for example, by eliminating redundant type declarations that are mandatory in Java as those can be evaluated by the compiler. Figure 6 shows the difference between variable declaration in Java and Kotlin.

Additionally, Kotlin allows you to reduce the number of variables in the code by introducing higher-order functions such as *also*, *apply*, *let* and *run*. Figure 7 demonstrates how *apply* function can be applied and its Java equivalent.

Such approach drastically reduces the number of variable declarations thus reducing the risk of having unused variables and may prevent several code smells, such as reusing variables that are used as method parameters.

Finally, it should be mentioned that running a simulation process is completely

---

[5]`https://kotlinlang.org/docs/reference/coroutines.html`

Figure 6. Variable declaration comparison with Java on the left and Kotlin on the right.

Figure 7. Demonstration of *apply* function application with its Java equivalent.

asynchronous. Kotlin provides various opportunities to execute code asynchronously but since we need a control over how many jobs are executed in parallel by the server, we decided to stick to a Java solution – a thread pool. This allows us to have a fixed number of threads that execute the simulation processes submitted by the users. If the amount of queued jobs is greater than the available thread count, the jobs are queued and executed as soon as a free thread is available. Such approach allows us to reduce the load on the server or, if the server has a lot of resources available, we can expand the thread pool to allow more concurrent jobs by editing a parameter in the configuration of the application.

4.2.3 Python

The core of the system is a predictive engine which relies on different machine learning techniques in order to provide an output for the end-user. For this task Python was the language of choice since it is often used for different scientific calculations due to ease of use and its vast variety of data processing libraries.

Python is an interpreted, object-oriented oriented programming language that combines remarkable power with clear and simple syntax [13]. It provides interfaces to different system calls as well as various window systems [13]. Since Python is an inter-

preted language, it is run by the interpreter and not compiled into bytecode or machine code by the compiler. One of the strengths of Python that it allows you to run code as soon as it is written, without waiting for the compile time.

Furthermore, Python uses high-level dynamic data types [13]. Having dynamic types means that variable type checking is only done to see that the type in the file matches the type of the receiving variable [3]. This allows programmers to write code faster, due to type declarations being redundant. However, this also means that errors are mainly discovered at runtime, as opposed to compile time for statically typed languages [3]. In order to solve this problem, Python introduced a module that adds support for type hints [13]. The aforementioned module gives programmers a possibility to detect type errors even before running the code if, for instance, they are using an integrated development environment.

As mentioned before, Python grants access to various libraries used for scientific calculations. Due to the project being heavily reliable on machine learning, the library named scikit-learn was used. Scikit-learn grants access to simple and efficient tools for data mining and data analysis [24]. In this project scikit-learn was mainly used as a provider for machine learning algorithms (for both classification and regression) such as random forest, gradient boosting and decision tree. Moreover, as scikit-learn presents various possibilities for data analysis and transformation, it was used as a tool to transform user data into a suitable format for machine learning algorithms.

4.3 Implementation

Primarily, there are two techniques used for delivering web content to the user. The first one, which is not so popular anymore, is delivering content to the user via web-pages. After clicking on a link another page is loaded and so on. The advantage of such approach is that the initial download may be smaller when accessing the web-page but this comes at the cost that every interaction requires a page reload.

The second way, the one that we used to develop our application, is a single page application (SPA). SPA might have a larger initial download size, but as the name suggests, interacting with a page does not cause the page reload. On user interaction, browser rerenders parts of the page, without requiring a page reload. Such approach improves user experience by providing responsive design and provides experience identical to the one when using a desktop application. In SPA, communication with the server is done via AJAX, which allows the server to respond to the events on the client side of the application.

Our application is built using the model-view-controller (MVC) approach. In such approach, model representation, user inputs and visualization part of the application are separated from each other and each task by different objects. The general approach is that by interacting with the controller (keyboard/mouse) user modifies the model which updates its representation in the view.

The model represents the data that is handled by the application. The model manages the behaviour of data, informs other components about its state and responds to the instructions to update its state [6].

The view is a graphical interface of the application, that handles state changes of the model and updates itself accordingly. Application usually have multiple views, even in SPA, which comes in a form of a single page but may have multiple views.

The controller interacts with the model by handling inputs from the user, usually from the keyboard or mouse, and commands the model the model the change its state and/or update the view to represent the new state of the model [6].

Now that we have discussed the general approach of the implementation, we can go into more detail and review how different parts of the application itself were implemented. It is important to note that the application is tightly connected to its configuration. Most of the parameters of the application can be changed through configuration and, in most cases, changes in configuration do not require the restart of the application, which means that the settings can be changed at runtime.

One of the main aspects of the application is to hide the complexity from the inexperienced users while still providing the advanced possibilities for those with a good grasp in machine learning and predictive process monitoring. With this in mind, we designed a control flow, displayed in figure 8.

Upon the first connection to the application, the user can choose whether a new log file should be uploaded or to continue with an already uploaded one. If the former is chosen, one has to follow the log uploading process, while if the latter is chosen, one is able to skip the upload process and continue with model training or model visualization if required model has already been trained before.

After the user has provided the system with the log file, the system analyzes the file in order to automatically detect case id, activity, timestamp and resource columns. Case id, activity and resource columns are detected by the name of the column. Names of the columns to look for are described in the configuration of the application and the options for the column names can be easily expanded. To find the timestamp column, the second row of the log file is read in addition to the log header. Then we try to parse each value with a date format specified in the configuration and if any of the parsing attempts succeeds, then the column is chosen to be the timestamp column.

When the previous step is completed, the user has to confirm that the system has classified the columns correctly or, if the system could not find the required columns, correct the selections and confirm the selection. Next, the system classifies the events into case and event attributes, which are also classified into categorical or numerical values.

An attribute is considered to be case attribute when its value does not change during the duration of a single case. On the other hand, an attribute is considered to be an event attribute, if its value changes over the case duration. Next, attributes can be classified into
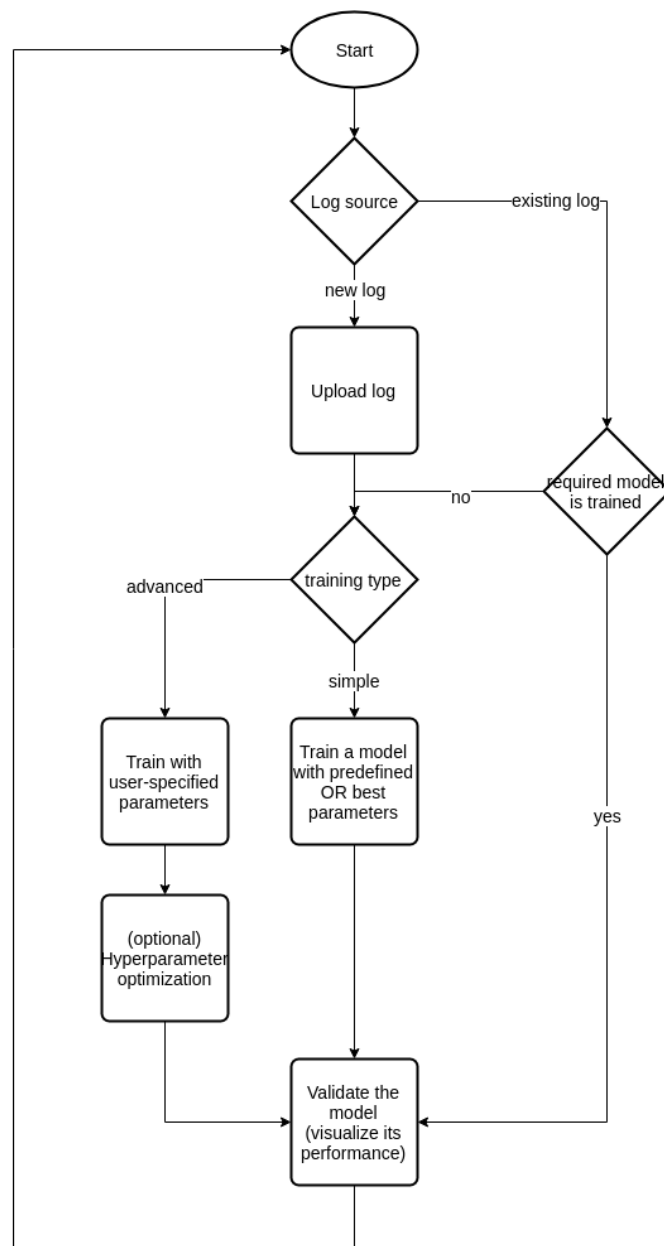
Figure 8. Control flow diagram for Nirdizati Training.

qualitative (categorical) or quantitative (numeric) attributes. An attribute is considered to be numeric if all of its values can be interpreted as an integer or a float and the amount of unique values is over a certain threshold. If contents of the attribute cannot be interpreted as a number of the number of unique numerical values is too small, an

attribute is considered be a categorical attribute.

After the classification is complete, the user has to confirm the classification. Once again user has an option to manually override the classification attributes selected by the system. This also includes the possibility to exclude an attribute from prediction, in which case the attribute will be ignored during the training process, or mark an attribute as an attribute that contains future values. Such attribute will be ignored during predictions, but the user is able to specify this attribute as a custom prediction target, in which case values for this attribute will be predicted. After the classification process is completed, the results are stored in the dataset descriptor file and written to disk. An example of a dataset descriptor file can be seen in figure 9.

```
{
    "static_cat_cols": [],
    "dynamic_cat_cols": [
        "Resource"
    ],
    "timestamp_col": "time",
    "activity_col": "activity_name",
    "static_num_cols": [
        "AMOUNT_REQ"
    ],
    "dynamic_num_cols": [
        "elapsed",
        "open_cases"
    ],
    "case_id_col": "case_id"
}
```

Figure 9. Example of a data set parameter file.

As a next step, the user is redirected to the training view, where he can trigger the training process. There are two training modes: basic, which is made to support the users not familiar with machine learning, and advanced mode, which allows manual construction of models.

The basic mode includes a set of predefined training parameters that will be used for prediction regardless of outcome target. Basic mode parameters are defined in the configuration of the application and can be changed on the runtime. It is also important to mention if a log file has optimized dataset parameters generated then those will be used for prediction, overriding the basic mode configuration.

If in the basic mode the only customizable thing is the prediction target, then advanced mode allows users to manually select machine learning algorithms that will be used in the prediction. In this mode, the user can manually select an encoding, bucketing and learner algorithms. Moreover, the user is able to specify hyperparameters that will be passed to the learner algorithm. A default value is provided for all hyperparameters but the user can override them manually to achieve better results.

```
<Node Identifier="n_estimators">
    <Value Identifier="control">org.zkoss.zul.Intbox</Value>
    <Value Identifier="max">400</Value>
    <Value Identifier="min">1</Value>
    <Value Identifier="default">300</Value>
</Node>
```

Figure 10. Property definition with a default value, maximum and minimum bounds.

Since hyperparameters are defined as properties of an algorithm in the configuration, it is possible to specify upper and lower bounds for the hyperparameters, which prevents the user from entering absurd values or creating simulations that would take too much time to complete. Figure 10 shows a definition of a property in a configuration file.

After the user has chosen the correct training parameters and specified hyperparameters, one can trigger the training process. When a simulation process is triggered, the job deployment process is started. Firstly, we have to generate an identifier for the job that user has submitted, so it does not interfere with the jobs submitted by other users. An id is generated using the MD5 hashing algorithm, which guarantees that the generated id will be unique for this job. Next, the job is deployed into a separate thread and submitted to the thread pool for the execution. From this step, execution of the job is asynchronous, which means that user can freely interact with the UI since this operation is non-blocking. When the thread is scheduled for execution, a training configuration file is generated based on the configuration chosen by the user. An example of such file can be seen in figure 11. Finally, the identifier of the job file is passed to the predictive engine, which then reads the parameters from the training configuration file, dataset descriptor file and based on those, trains a model.

```
{
  "remtime": {
    "cluster": {
      "laststate": {
        "gbm": {
          "n_clusters": 2,
          "max_features": 0.5,
          "n_estimators": 300,
          "learning_rate": 0.1
        }
      }
    }
  },
  "ui_data": {
    "start_time": "2018-03-31T12:17:27.994Z",
    "log_file": "logdata/BPI2012A.csv",
    "job_owner": "4822091944800935abcdb769240df29e"
  }
}
```

Figure 11. Example of a training parameter file.

On training model completion, the user can navigate to the validation view of the model. Here, the user can see different visualizations of various model performance metrics. This allows the user to validate the accuracy of the model. If the user is satisfied with the trained model, the model can be downloaded as a *pickle* file. Furthermore, the user can also export the results files that are used for the visualization, that were generated by the predictive engine. The user can export a single file or export all result files as a *zip* archive. However, if the user is not satisfied with the outcome, one can train a new model using different algorithms and/or hyperparameter values.

5 Running example

In this section, a walkthrough for the application is presented from a point of view of a user who is a first-time visitor to the application.

Firstly, a user has to connect to the application via a modern browser that is compatible JavaScript and ZK framework[6]. After the connection has been established, the user is presented with two choices, shown in figure 12, to either upload a new log or to continue with existing one. The former option exists for the users that have already been through the upload process. From here, the user clicks 'upload new log' button and thus, is redirected to the log upload page.
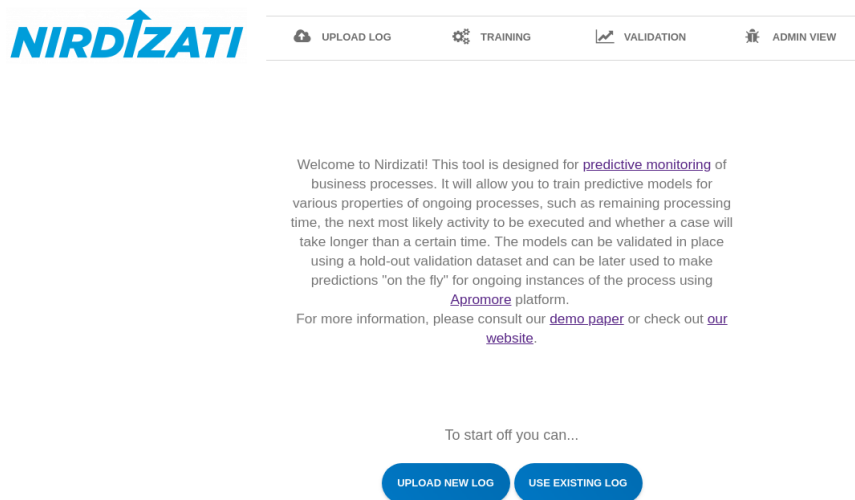
Figure 12. Nirdizati Training landing page.

Figure 13 depicts the view that is opened to the user after being redirected to the upload page. Here the user has two possibilities to upload a log. Firstly, the log can be dragged and dropped into the grey container from the users' filesystem. The drag and drop functionality is also supported for the file upload.

Once the user has chosen the file from the filesystem, it is analyzed by the server. If the extension of the file is found to be suitable (e.g. *CSV* extension) then file name appears in the grey container to indicate the user that the file is with the correct extension and that the user can proceed with the next action (figure 13). However, if the user tries to upload file with an extension that different from the allowed one, then the container turns red and user is notified with an error message, shown in figure 14.

---

[6]`https://www.zkoss.org/whyzk/Features#note`

Figure 13. Nirdizati Training upload page showing that file with correct file extensions has been selected.
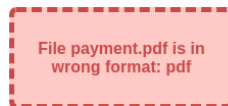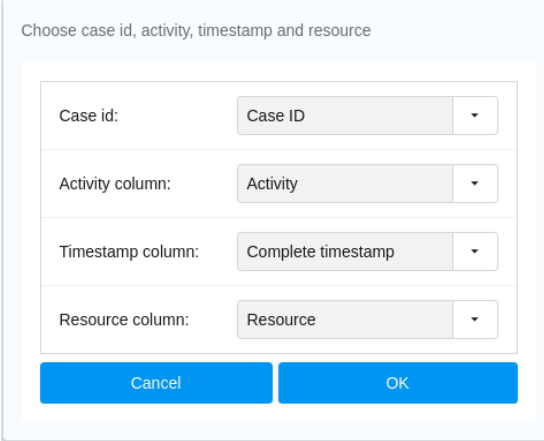

Figure 14. File with incorrect extension is uploaded.

The next step is to generate dataset parameters for the log. In order to achieve this, the user has to click the 'create dataset parameters' button. After the user has clicked the button, server parses the uploaded file in order to automatically identify the required columns from the users' file such as case identifier, activity column, resource column and the timestamp column. When the classification of columns has been completed, the user is presented with a window shown in figure 15. In this window, one can see automatically categorized columns and, if the system has classified the columns incorrectly, the user is allowed to select the correct value for aforementioned columns. When the user is

30

satisfied with the result, 'OK' button should be clicked in order to continue the process. At any time during this process user can click the 'cancel' button in order to stop the generation process. If the user decides to do so, the log file will not be saved to the server.

Figure 15. The window showing automatically detected case id, activity, resource and timestamp columns.

Next, all the other data columns that are contained in the log should be classified as case and event attributes, which in turn are divided into numerical or categorical values. When the server is done with categorization, the user is shown a window similar to the one in figure 15, which displays column name on the left and its category on the right. Once again user can correct the system if categorization was faulty. Finally, the user has to click the 'OK' button to finish dataset parameter generation or cancel the process by clicking the 'cancel' button.

After the user confirms the parameters, a notification is displayed in the bottom left corner, indicating that the process was successfully finished and that the user will be redirected to the basic training view. Figure 16 displays the training view that user is redirected to. From here, the user can select the prediction target by selecting a value in the combo box. If user selects case duration as the prediction target, then the threshold can also be modified by the user. To start the simulation process, one should click the 'train model' button.

When the simulation process is started, on the right-hand side a new component is added to the user interface. The purpose of the component shown in figure 17, is to help the user to track the started simulations. From here the user can see the name of the log file, prediction target, the status of the simulation (whether it is running, failed, completed or waiting to be started in the queue), as well as model parameters and hyperparameters that were chosen for the simulation.

Finally, one has to wait for the simulation completion, but since the simulations are

Figure 16. Nirdizati Training training view.

Figure 17. The component that helps the user to track started simulations.

non-blocking operations, the user can start more simulations with different prediction targets or even upload a different log file.

In order to notify the user of process completion, a notification is shown in the left bottom corner of the screen. Furthermore, the user can now click the 'validate' button in the tracker component in order to be redirected to the model validation view. Please note that clicking the 'validate' button in the tracker component is equivalent to pressing the 'Validation' tab in the header and then selecting the same job in the table.

This gives one access to the next step, validation view. In this view, the user can see meta data that is associated with the simulation as well as review model accuracy by visualizing different aspects of the model.

Firstly, the user can view the model accuracy by comparing actual values in the log versus the values that the model has predicted. This is done in the 'Actual vs predicted'

tab. For the regression model, the chart is shown as a scatter plot (figure 18). However, for classification models given chart is represented as a confusion matrix (figure 19).
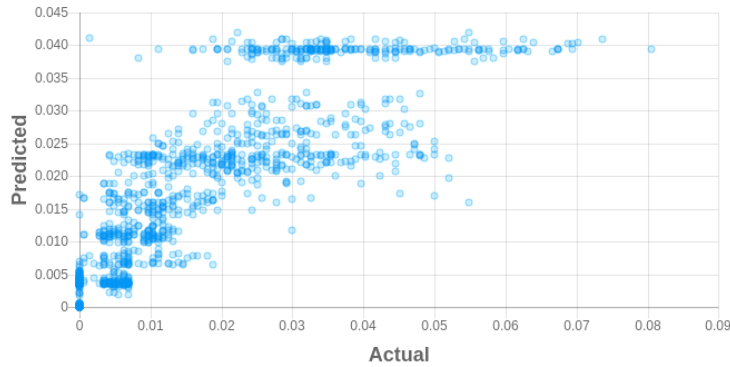

Figure 18. Scatter plot: actual (x-axis) vs predicted (y-axis) shown for regression.
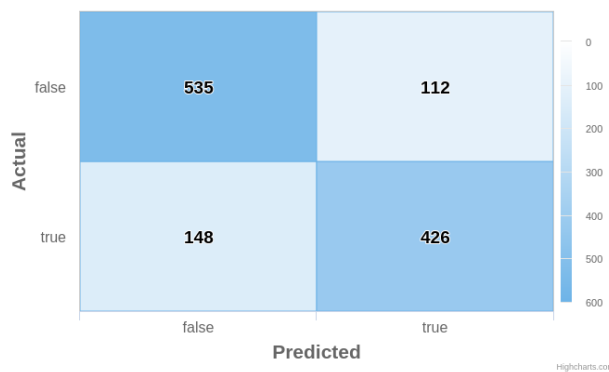

Figure 19. Confusion matrix: actual (x-axis) vs predicted (y-axis) shown for classification.

Secondly, on the 'Accuracy vs prefix length' tab, one can check the model accuracy using a range of evaluation metrics depending on whether it is a regression or a classification problem. Since we validate the model after each executed event, each point on the chart shows average accuracy after each executed event for given prefix length. For example, if on the x-axis the number of events is 2, then predictions for all ongoing prefixes with length 2 have been made. On the y-axis, an average accuracy for all prefixes with length 2 is displayed.

33

As depicted in figure 20, all charts are represented as line charts on this page. Not only can one evaluate the accuracy of a single model on this page, but the user can also compare given models accuracy with other relevant models. The relevant model can be defined as a model that has been trained using the same log file and predicted the same outcome target. Comparison of two different models can be seen in figure 21.
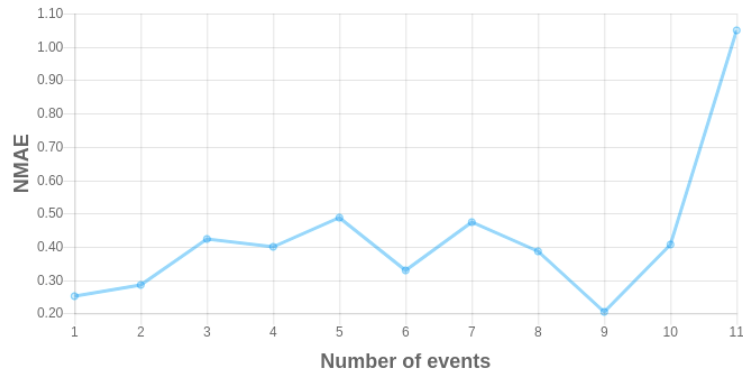

Figure 20. Line chart: number of events (x-axis) versus normalized mean absolute error (y-axis)
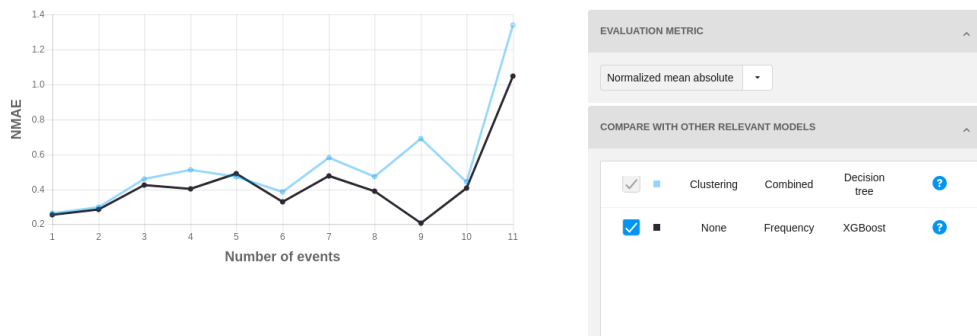
Figure 21. Accuracy comparison between two models (normalized mean absolute error)

Lastly, in the last tab, named 'Feature importance', one can see the weight of each feature for given model configuration as a bar chart (figure 22). Feature importance is evaluated in terms of mean decrease impurity available in *sklearn*. In simple words, feature importance shows how much the accuracy of a given model decreases if we drop or assign random values to this feature.

Execution Times All test cases were conducted using Kotlin version 1.2.31 on a desktop machine with 3,3GHz Intel i5-6500 processor and 16GB of DDR4-2333MHz

34

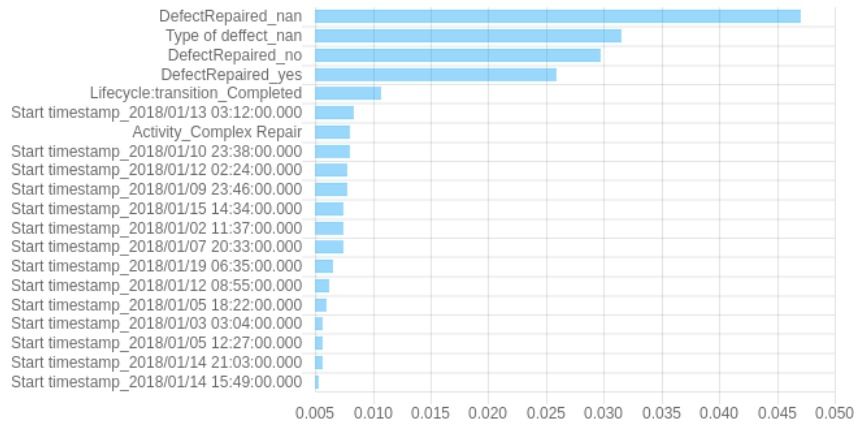Figure 22. Feature importance chart: feature weight (x-axis) vs feature name (y-axis).

RAM. For the test cases, event log named 'repairExample2_ENG.csv' which is provided as an attachment to the thesis. The log contains 1048576 events and 10 columns. Time was measured before entering the function that is responsible for providing the required information and after the aforementioned function has provided the result. The difference of those timestamps shows us the time required for the task completion.

Table 1 shows average time required to automatically detect the case identifier, activity, resource and timestamp columns.

Table 1. Column auto detection times for 'repairExample2_ENG.csv' log.

Try n	Time to parse (ms)
1	3
2	0
3	1
4	3
5	0
6	1
7	0
8	4
9	1
10	1
Average:	1.4

Table 2 shows average time required to classify the data attributes in each column in the log into categorical and numerical case or event attributes.

35

Table 2. Column classification times for 'repairExample2_ENG.csv' log.

Try n	Time to parse (ms)
1	601
2	582
3	516
4	504
5	499
6	566
7	510
8	497
9	494
10	527
Average:	529.6

Table 3 shows the time required to parse the result files provided by the predictive engine, in order to provide the visualizations to the user. In practice, result parsing is done only once for each model, since the parsed results are cached. This means that when reopening the view for an already cached model, we do not need to perform any IO operations.

Table 3. Times to parse the evaluation results for 'repairExample2_ENG.csv' log.

Try n	Time to parse (ms)
1	19
2	15
3	17
4	20
5	13
6	18
7	4
8	14
9	4
10	4
Average:	12.8

6 Summary

In this thesis, we have developed a web application that makes it easier for the end users to train, evaluate and compare predictive models for the online business process monitoring. The application allows users to quickly upload their event log, construct the model by providing a set of baseline parameters, which provide the most accurate result in most cases, and then export the model to use it for live predictions on an event stream or to analyze the model further using third-party applications. Moreover, more experienced users can use the application in the advanced mode, which allows users to fine-tune many settings, such as the choice of a machine learning algorithm and its core hyperparameters that will be used for learning.

As a result, we have developed a SaaS solution for predictive process monitoring. All of the computations are done on the server side and the application allows users simple access via web browser. This means that user does not need powerful hardware to train the models and analyze the data. Furthermore, user models and result files are stored on the server, thus allowing easy access at any time if the internet connection is present.

The developed application is very configurable, which means that most of the settings can be changed without recompiling the application. Furthermore, as the configuration is read from the external file, a lot of the application settings can be changed at runtime of the application, without requiring a restart. Compiled web application comes in a *web application archive (WAR)* format, which allows it to run in any Java container, such as Tomcat or Jetty. Such approach does not enforce any platform limitations or hardware requirements.

The source code has been released as an open source software at `https://github.com/zukkari/nirdizati-training-ui`. The developed engine has been deployed to the server and can be accessed at `http://training.nirdizati.com`.

7   Future work

There are some clear points of improvement for developed software. Firstly, *XES* support should be added. As mentioned in subsection 2.1, this format is widely used in process mining and was designed to describe business processes. This would allow users to upload the log directly to the application without the need of using an external converter.

Secondly, a database integration could be added. Currently, all of the data is stored on the disk, which requires a lot of manual IO operations, such as looking for the correct files in the correct folders. A database integration would allow easier data manipulation and also add a possibility where the data could be stored remotely (e.g. on the different machine from the one where the UI instance is running on.)

Furthermore, database integration would allow to implement better user authentication and therefore authorization. Currently, no authentication mechanism is implemented and users are distinguished via cookies. Database would allow us to implement username/password authentication and thus expose some features that would be only accessible by authorized users. One of such features is hyperparameter optimization, which is a very CPU intensive task.

References

[1] IEEE standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84, Dec 1990.

[2] IEEE standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50, Nov 2016.

[3] M. Abadi, L. Cardelli, B. C. Pierce, and G. D. Plotkin. Dynamic typing in a statically typed language. *ACM Trans. Program. Lang. Syst.*, 13(2):237–268, 1991.

[4] A. Breslav. Kotlin 1.0 released: Pragmatic language for jvm and android. `https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/`, Feb. 2016. Accessed: 11.03.2018.

[5] A. W. Brown. *Trends in Software Engineering*, volume 54. Academic Press, 1 edition, 2001. Page: 4.

[6] S. Burbeck. Applications programming in smalltalk-80(tm): How to use model-view-controller (mvc), 1987.

[7] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 13-17, 2016*, pages 785–794. ACM, 2016.

[8] M. de Leoni, W. M. P. van der Aalst, and M. Dees. A general framework for correlating business process characteristics. In S. W. Sadiq, P. Soffer, and H. Völzer, editors, *Business Process Management - 12th International Conference, BPM 2014, September 7-11, 2014. Proceedings*, volume 8659 of *Lecture Notes in Computer Science*, pages 250–266. Springer, 2014.

[9] T. G. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In C. E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004)*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

[10] M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.

[11] J. Evermann, J. Rehse, and P. Fettke. XES tensorflow - process prediction using the tensorflow deep-learning framework. In X. Franch, J. Ralyté, R. Matulevicius, C. Salinesi, and R. Wieringa, editors, *Proceedings of the Forum and Doctoral*

Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering, CAiSE 2017, volume 1848 of *CEUR Workshop Proceedings*, pages 41–48. CEUR-WS.org, 2017.

[12] M. Federici, W. Rizzi, C. D. Francescomarino, M. Dumas, C. Ghidini, F. M. Maggi, and I. Teinemaa. A ProM operational support provider for predictive monitoring of business processes. In F. Daniel and S. Zugal, editors, *Proceedings of the BPM Demo Session 2015 Co-located with the 13th International Conference on Business Process Management (BPM 2015)*, volume 1418 of *CEUR Workshop Proceedings*, pages 1–5. CEUR-WS.org, 2015.

[13] P. S. Foundation. Python 3.6.5rc1 documentation. `https://docs.python.org/3/index.html`, Mar. 2018. Accessed: 17.03.2018.

[14] J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. *The Java Language Specification*. Java se 8 edition edition, Feb. 2015. Chapter 4. Types, Values and Variables.

[15] C. W. Günther and E. Verbeek. *XES Standard Definition*. 2014.

[16] JetBrains. Company - jetbrains. `https://www.jetbrains.com/company/?fromMenu`. Accessed: 11.03.2018.

[17] JetBrains. Null safety - kotlin programming language. `https://kotlinlang.org/docs/reference/null-safety.html`. Accessed: 11.03.2018.

[18] JetBrains. Sealed classes - kotlin programming language. `https://kotlinlang.org/docs/reference/sealed-classes.html`. Accessed: 11.03.2018.

[19] JetBrains. Faq - kotlin programming language. `https://kotlinlang.org/docs/reference/faq.html`, June 2017. Accessed: 11.03.2018.

[20] K. Jorbina. A web-based tool for predictive process analytics. Master's thesis, University of Tartu, 2017.

[21] A. Liaw and M. Wiener. Classification and regression by randomforest. 23, 11 2001.

[22] B. L.P. Potix corporation: Private company information - bloomberg. `https://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapId=30517079`. Accessed: 10.03.2018.

[23] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[24] 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. *CoRR*, abs/1201.0490, 2012.

[25] Potix. Zk - zuml reference/zuml/languages/zul - documentation. `https://www.zkoss.org/wiki/ZUML%20Reference/ZUML/Languages/ZUL`. Accessed: 10.03.2017.

[26] Potix. Zk framework. `https://www.zkoss.org/product/zk`. Accessed: 10.03.2018.

[27] A. Rozumnyi. A dashboard-based predictive process monitoring engine. Master's thesis, University of Tartu, 2017.

[28] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[29] I. Verenich. A general framework for predictive business process monitoring. In O. Pastor, S. Rinderle-Ma, R. Wieringa, S. Nurcan, B. Pernici, and H. Weigand, editors, *Proceedings of CAiSE 2016 Doctoral Consortium co-located with 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016)*, volume 1603 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[30] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and I. Teinemaa. Survey and cross-benchmark comparison of remaining time prediction methods in the context of business process monitoring. *Preprint available at https://arxiv.org/abs/1805.02896*, 2018.

[31] I. Verenich, S. Mõškovski, S. Raboczi, M. Dumas, M. L. Rosa, and F. M. Maggi. Predictive process monitoring in Apromore . 2018. Accessed: 30.03.2018.

[32] Z.-H. Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition, 2012.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Stanislav Mõškovski**,

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 framework for the evaluation of predictive process monitoring techniques

   supervised by Ilya Verenich

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, 12.05.2018