

UNIVERSITY OF TARTU  
Institute of Computer Science  
Software Engineering Curriculum

**Ayham Taleb**

# **A Web Tool For The Comparison Of Predictive Process Monitoring Algorithms**

**Master's Thesis (30 ECTS)**

Supervisors: Fabrizio Maria Maggi, PhD,  
Fredrik Payman Milani, PhD

Tartu 2017

## **A Web Tool For The Comparison Of Predictive Process Monitoring Algorithms**

### **Abstract:**

Predictive Process Monitoring analyzes an event log aiming to predict critical business metrics as time, cost and process outcomes. Various techniques and approaches of predictions were developed in both academia and industry sectors in order to provide understandable predictions to the users. In this Master's Thesis, we introduce a web based tool for the comparison of predictive process monitoring algorithms which provides researchers or end users involved in this field an easier way for choosing the suitable prediction approach to a certain log. This project uses a queuing system which is able to build different predictive models at the same time. We show the results of different predictive models with a visual comparison that allows the evaluation of each predictive model. The new functionalities have been implemented in a web application, which allows users to configure and trigger the tasks of the queuing system and shows the results. The application has been evaluated on a real-life log pertaining to the treatment process of sepsis patients in a hospital.

### **Keywords:**

Predictive Process Monitoring, Process Mining, Process Analytics Tool

**CERCS: P170**

## **Veebipõhine tööriist võrdlemaks prognoositavate algoritmiseirete protsesse**

### **Lühikokkuvõte:**

Ennetava Protsessi Jälgimine analüüsib sündmuste logi, mille eesmärk on prognoosida kriitilisi ärimõõdikuid aja, kulude ja protsessi tulemuste põhjal. Mitmed ennustamise tehnikad ja lähenemisviisid on välja töötatud nii akadeemilises kui ka tööstussektorites, et pakkuda kasutajatele arusaadavaid ennustusi. Selles magistritöös tutvustame veebipõhist tööriista, et võrrelda prognoositavate algoritmiseirete protsesse. See pakub teadlastele või selles valdkonnas olevatele lõppkasutajatele lihtsamaid viise valimaks kindlale logile sobivat prognoosimisviisi. See projekt kasutab järjestavat süsteemi, mis suudab samal ajal luua erinevaid prognoosimudeleid. Näitame erinevate prognoosimudelite tulemusi kasutades visuaalset võrdlust, mis võimaldab hinnata iga ennustatavat mudelit. Uued funktsioonid on seadistatud veebira-kenduses, mis võimaldavad kasutajatel seadistada ja käivitada järjestava süsteemi ülesandeid ja seejärel näitab tulemusi. Rakendust on hinnatud päriselu logi põhjal, mis on seotud haiglas olevate sepsisehaigete patsientide raviprotseduuriga.

### **Võtmesõnad:**

Protsessi Ennustav Seire, Protsessikaeve, Protsessi Analüüsitööriistad

**CERCS: P170**

## Table of Contents

1	Introduction .....	5
2	Related Work .....	6
2.1	Remaining Time Predictions .....	6
2.2	Outcome Based Predictions .....	6
3	Background .....	7
3.1	Process mining .....	7
3.1.1	Event Logs .....	7
3.2	Predictive Process Monitoring .....	8
3.2.1	Encoding Methods .....	9
3.2.2	Prediction Methods .....	11
3.2.3	Clustering methods .....	12
3.3	A Web-Based Tool For Predictive Process Analytics .....	12
3.3.1	General Framework .....	12
3.3.2	Back-end Application .....	13
3.3.3	Front-end Application .....	13
4	Contribution .....	15
4.1	Framework for The Comparison of Predictive Process Monitoring Algorithms .....	15
4.1.1	The Implemented Methods .....	15
4.2	Types of Encoding .....	16
4.3	Clustering .....	16
5	Tool Implementation .....	18
5.1	Front-end Application .....	18
5.1.1	Views in the Application .....	18
5.2	Back-end Application .....	22
5.3	Queuing System .....	22
5.3.1	Classification Configuration .....	23
5.3.2	Regression Configuration .....	24
6	Evaluation and Verification .....	25
6.1	Evaluation Measures .....	25
6.1.1	Root Mean Square Error .....	25
6.1.2	Mean Absolute Error .....	25
6.1.3	Coefficient Of Determination .....	25
6.1.4	Accuracy .....	25

6.1.5 Area Under The Curve .....	26
6.1.6 F1 Score .....	26
6.2 Results .....	26
6.2.1 Regression Results .....	26
6.2.2 Classification Results .....	30
7 Conclusion .....	35
8 Acknowledgement .....	36
References .....	37

# 1 Introduction

Nowadays, several companies store process execution information in the form of textual files. These files are called *event logs*, which hold historical traces of completed process executions [17]. The logs are the entry point of any process mining techniques. Process mining allows one to gain advantage of past process executions, to discover the flow of the activity executions (process discovery), check the conformance of a log by performing a replay against an existing business process model, and to enhance the processes by using the information available in the log.

In the context of process mining a family of techniques based on providing continuous estimations on future process executions can be found, called Predictive Process Monitoring. It provides the user with continuous predictions in a currently running process case in order to achieve some business goals. Predictive Process Monitoring builds a predictive model from historical traces. At runtime partial traces of a currently running case are used to query the predictive model and derive predictions about the achievement of a goal in the future development of the case. Example goals are discovering any defected cases, missed deadlines or anything which might go wrong in the process.

Currently in both academia and industry sectors there is a number of approaches used in Predictive Process Monitoring for the creation of the predictive model. Choosing the suitable approach for a given log can be a time consuming task, and there is no tool support to provide a visual comparison among different approaches.

In this thesis we aim to answer the following questions, on a given log: “What is the suitable Encoding method?”, “Do clustering adds any value to the predictive model?”, “In case of remaining time predictions which is the most suitable regression method?” and “Which classifier should be used in the outcome based predictions?”.

In order to provide answers to the mentioned question, we have implemented a Web Tool For The Comparison Of Predictive Process Monitoring Algorithms. The tool allows the user to build different predictive models at the same time. We show the results of different predictive models with a visual comparison that allows the evaluation of each of them. The application has been evaluated on a real-life log pertaining to the treatment process of sepsis patients in a hospital.

## 2 Related Work

### 2.1 Remaining Time Predictions

The approaches that study time predictions answer questions as “How long does it take for an activity A to be completed?”, “In a running case, will activity B be performed in a certain time” and “What will be the duration of a currently running case?”.

In [1] the authors presented a regression based method for remaining time prediction, using information about activities’ durations, occurrence and the activities’ attributes. The approach considered the duration as a continuous variable, the occurrences are ordered and the activity’s attributes unordered variables. The evaluation of the presented approach showed that their regression based method performs better than the naive approach of averaging remaining times. In [2] the authors have introduced an approach based on building a transition system by using abstraction mechanism on the past executions, the transition system is annotated by attaching a set of measurements to each state. The annotated system is used to derive predictions. On the other hand in [9], the authors predicted the remaining execution time of a process using stochastic Petri nets.

### 2.2 Outcome Based Predictions

Outcome predictions aim to classify the running cases to predict the outcomes in business processes. This type of prediction aims to answer several questions related to the ongoing cases, for example, “Do activity A always happen before activity B in the ongoing trace?”, “Given a specified time do the ongoing trace complete within that time?” and “Do the ongoing trace result in a negative outcome?”.

A framework presented in [3] provides predictions on the outcome of an ongoing trace using Decision Tree Learning to predict the fulfilment of Linear Temporal Logic (LTL) rules. In [4] a tool for outcome prediction is presented. The tool gives the user different methods for clustering and classification. The user would have to set the prefix length, select either Decision Tree or Random Forest Learning method after selecting a clustering method. The tool provides Model-based clustering and the DBSCAN clustering. Another tool [5] predicts if the ongoing trace will be completed “on time” or “late”. The tool allows the user to use only Random Forest Learning or combine the learning method with a clustering method. The evaluation of the tool has shown that the predictions of a combination of Random Forest Learning and k-medoids Clustering outperforms others. [10, 11] aimed to predict the next task, in [10] the authors used a similar transition system used in [2] to predict the next sequences of tasks. [11] used Long Short-Term Memory (LSTM) Neural Networks to predict the next sequences of tasks.

## 3 Background

### 3.1 Process mining

Discovering, monitoring and providing improvements to the business processes are exactly the aim of Process Mining. Process mining entry point is an event log. The event log contains cases (traces). Each case is a sequence of events (activities). An event has information about the execution of an activity, a timestamp, in addition to information about the data used during the execution of the activity. Beside the execution related information, an event can hold information about who has executed an activity (actors) [6].

There are three types of operations used in process mining on an event log: Process Discovery, Conformance Checking, and Enhancement. Process discovery, takes an event log as input and aims to discover a model from the event log. The models can be represented using UML activity diagram, BPMN or Petri net. For Conformance checking in addition to the log a model is given as input. The primary objective of conformance checking is to validate if the given model is compliant with the log. Enhancement also requires a log with a model. Enhancement produces a new improved model based on information retrieved from the given log. [6]

#### 3.1.1 Event Logs

An event log is the entry point of Process Mining. The logs used in this project are in Extensible Event Stream (XES) [7] format. A XES file follows XML schema, the root is the log, the log contains the traces, the trace tag holds the events and information about the trace itself (trace attributes). The event has a timestamp, an id and additional attributes related to the event. An example of log is shown in Figure 1. An event log  $L$  is a set of traces, where  $\alpha_c = (e_c^1, e_c^2, \dots, e_c^n) \in \Sigma^*$  represents a trace,  $\Sigma^*$  a set of all possible strings over the alphabet  $\Sigma$ ,  $c$  is the case id, and  $n$  is the size of case. An event  $e_c = (a, t, d_1, \dots, d_m)$ , where  $a \in \Sigma$  is the process activity associated to the event,  $t \in N$  is the event timestamp and  $d_1, \dots, d_m$  is a list of additional attributes (event attributes) [10].

```

<string key="lifecycle:transition" value="complete"/>
<string key="concept:name" value="Release A"/>
<date key="time:timestamp" value="2014-12-16T17:00:00.000+01:00"/>
</event>
</trace>
<trace>
<string key="concept:name" value="LNA"/>
<event>
<boolean key="InfectionSuspected" value="false"/>
<string key="org:group" value="L"/>
<boolean key="DiagnosticBlood" value="false"/>
<boolean key="DisfuncOrg" value="false"/>
<boolean key="SIRSCritTachypnea" value="false"/>
<boolean key="Hypotensie" value="false"/>
<boolean key="SIRSCritHeartRate" value="false"/>
<boolean key="Infusion" value="false"/>
<boolean key="DiagnosticArtAstrup" value="false"/>
<string key="concept:name" value="ER Registration"/>
<int key="Age" value="50"/>
<boolean key="DiagnosticIC" value="false"/>
<boolean key="DiagnosticSputum" value="false"/>
<boolean key="DiagnosticLiquor" value="false"/>
<boolean key="DiagnosticOther" value="false"/>
<boolean key="SIRSCriteria20rMore" value="false"/>
<boolean key="DiagnosticXthorax" value="false"/>
<boolean key="SIRSCritTemperature" value="false"/>
<date key="time:timestamp" value="2014-12-03T10:50:28.000+01:00"/>
<boolean key="DiagnosticUrinaryCulture" value="false"/>
<boolean key="SIRSCritLeucos" value="false"/>
<boolean key="Oligurie" value="false"/>
<boolean key="DiagnosticLacticAcid" value="false"/>
<string key="lifecycle:transition" value="complete"/>
<boolean key="Hypoxie" value="false"/>
<boolean key="DiagnosticUrinarySediment" value="false"/>
<boolean key="DiagnosticECG" value="false"/>
</event>
<event>
<string key="org:group" value="C"/>
<string key="lifecycle:transition" value="complete"/>
<string key="concept:name" value="ER Triage"/>
<date key="time:timestamp" value="2014-12-03T10:54:19.000+01:00"/>
</event>
<event>
<string key="org:group" value="L"/>
<string key="lifecycle:transition" value="complete"/>
<string key="concept:name" value="ER Sepsis Triage"/>
<date key="time:timestamp" value="2014-12-03T10:54:39.000+01:00"/>
</event>
</trace>
</Log>

```

Figure 1: XES file of the Sepsis patients Log

## 3.2 Predictive Process Monitoring

Predictive Process Monitoring is a family of techniques providing continuous estimations, predictions, and recommendations to the user to achieve business goals in a currently running process case. Predictive Process Monitoring takes an event log and builds a predictive model based on the historical process execution cases. The model is used to continuously provide the user with predictions over running cases. Various Predictive Process Monitoring techniques have been developed which can be classified based on the type of prediction they provide for example, remaining time prediction, and outcome based predictions [3]. Building a predictive model always takes an event log as input. The log is encoded and different combinations of clustering, prediction methods are applied on the encoded log to achieve a specific type of predictions. Figure 2 is an overall description of Predictive Process Monitoring.



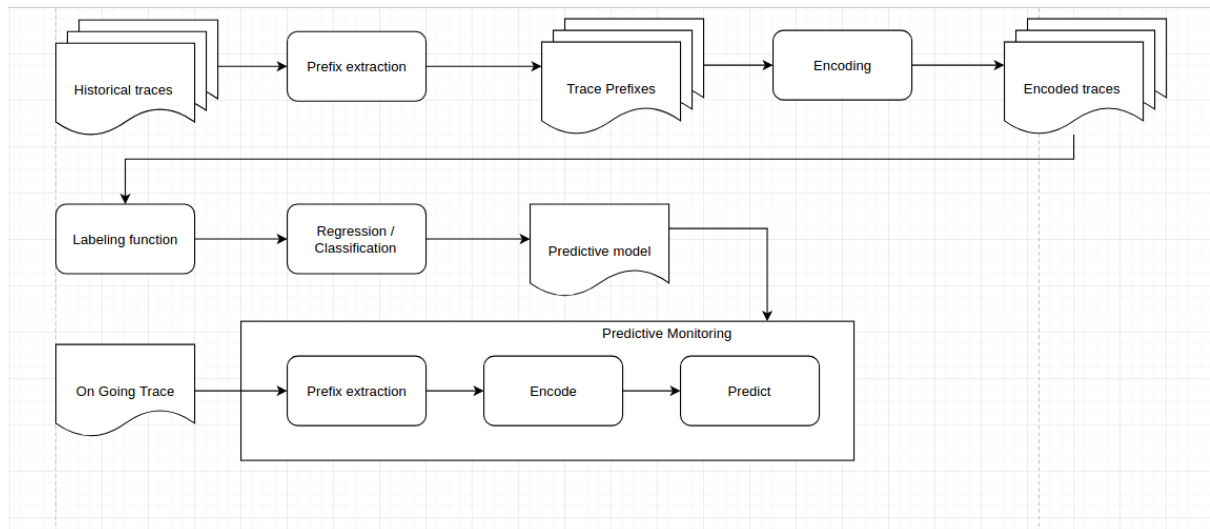


Figure 2. Predictive process monitoring

Figure 3 shows a Predictive Process Monitoring approach combined with clustering.

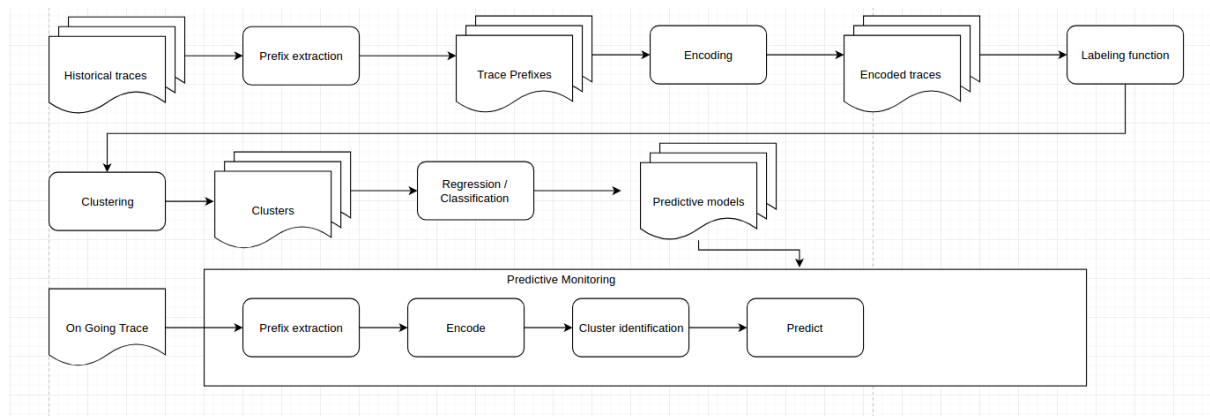


Figure 3. Predictive process monitoring combined with clustering

In Figure 2 and 3, both approaches have a similar flow. They start by extracting the prefixes of the historical traces, which leads to a log that contains prefixes with length equals to the given prefix length. Then the encoder encodes the traces prefixes. Once the encoder job is done each trace is labelled. The labelled traces are the input of the regression/classification method which will output the predictive model. The predictive model is stored to be used on ongoing encoded traces. The approach given in Figure 3 differs from the one in Figure 2 because the encoded traces are the input of a clustering method. The labels of the traces are dropped before clustering. The clustering will output clusters. These clusters are the input of the regression/classification method which will output one predictive model per cluster. At runtime we identify to which cluster an ongoing trace belongs, and that trace is forwarded to the predictive model corresponding to that cluster.

### 3.2.1 Encoding Methods

The encoded log is the log which will be feed to the algorithms to create the predictive models. For us to achieve a comparison tool we have looked and implemented most of the encoding methods presented in [8].

**Index-Based Encoding:** Each trace is represented as a sequence of events, the data associated to the event are divided into static and dynamic information. The static information is the same for all the events in the sequence (e.g., information related to the case), dynamic information is event related such as timestamp, the event Id and other event attributes. The case vector is represented as:  $t_i = (s_i^1, \dots, s_i^u, event_{i1}, event_{i2}, \dots, event_{im}, h_{i1}^1, h_{i2}^1, h_{im}^1, \dots, h_{i1}^r, h_{i2}^r, h_{im}^r)$ , Where  $s_i^1, \dots, s_i^u$  are the static attributes, each  $event_{ij}$  is the event at position  $j$ ,  $h_{ij}$  is the dynamic feature at position  $j$ . Figure 4 shown an example of *Index-Based Encoding*.

	age	event_1	...	event_m	...	department_1	...	department_m	label
$\sigma_1$	33	consultation		ultrasound		radiotherapy		nursing ward	false
...									
$\sigma_j$	56	order rate		payment		general lab		clinic	true

*index-based encoding.*

Figure 4. *index-based encoding* [8]

**Simple Index-Based Encoding:** Each trace is a sequence of events, differently from *Index-Based Encoding*, the events are represented to show only the control flow of a case and an event holds a unique dynamic attribute. The trace case be presented as:  $t_i = (event_{i2}, \dots, event_{im})$ , where  $event_{ij}$  is the event id or name at position  $j$ .

**Boolean Encoding:** In this method, the trace is presented as a vector of booleans. Each element of the vector corresponds to an event and is True if the event occur in the trace False otherwise.

**Frequency-Based Encoding:** In this method, the trace is represented as a numeric vector. Each element of the vector is the number of occurrences of an event in the trace.

**Index Latest Payload Encoding:** Is similar to *Index-Based Encoding*. The difference is that only the attributes of the latest event occurred in the trace are taken into consideration.

Figure 5 shows an example of the last four encoding methods.

	consultation	ultrasound	...	payment	label
$\sigma_1$	1	1	...	0	false
...					
$\sigma_k$	0	0	...	1	true

(a) *boolean encoding.*

	consultation	ultrasound	...	payment	label
$\sigma_1$	2	1	...	0	false
...					
$\sigma_k$	0	0	...	4	true

(b) *frequency-based encoding.*

	event_1	...	event_m	label
$\sigma_1$	consultation		ultrasound	false
...				
$\sigma_k$	order rate		payment	true

(c) *simple index encoding.*

	age	event_1	...	event_m	...	department_last	label
$\sigma_1$	33	consultation		ultrasound	...	nursing ward	false
...							
$\sigma_k$	56	order rate		payment	...	clinic	true

(d) *index latest payload encoding.*

Figure 5. *Encoding Methods* [8]

### 3.2.2 Prediction Methods

For us to build a predictive model, the encoded log has to be passed to a learning method. The learning methods used in this project can be grouped into two groups, Classification and Regression.

1. **Classification** is an approach for building predictive models from a training dataset to predict categorical outcomes. The aim of classification technique is to categorize new records using the same features of the training dataset. There are multiple classifiers for solving classification problems like Decision Trees, Random Forest, K-Nearest Neighbor (KNN), Rule-Based Classifier, Naïve Bayes Classifiers etc. In this subsection we will focus on the classifiers used in the tool we have implemented in this project.
  - a) **Decision Trees** A decision tree is a flowchart like in a shape of tree, it has leaves and branches. The decision tree assigns a probability to each of the possible leaf based on the context. The input is an encoded log, the output is a predictive model which can be used to predict the outcome of ongoing encoded traces.
  - b) **Random Forest** Is a combination of decision trees, such that each tree depends on the values of a random independent vector with the same distribution of all the trees in the forest. The error of a forest classifiers depends on the strength of each tree in the forest and the relation between them [13]. The creation of multiple decisions tree on a dataset improves the accuracy of the prediction model.
  - c) **K-Nearest Neighbor (KNN)** Is an non parametric lazy learning algorithm. Non parametric means that KNN does not make any assumptions about the distribution of the dataset. By lazy we mean that KNN generalization is not relying on the attributes of the input dataset. Simply KNN provides a prediction based on the nearest neighbor in the dataset.
2. **Regression** is a machine learning technique which predicts a number based on an estimated function from the given dataset. To train a regression model, the dataset last column is the target or the expected value to be predicted. We have chosen the following regression methods to be included in our tool.
  - a) **Linear Regression** Fits a linear function based on independent variables in the dataset. If the dataset has only a single feature then it is defined as Simple Linear Regression. If the dataset has multiple features then it is defined as multivariate Linear Regression. The function can be presented in this formula  $y=f(x)$  where  $f(x)$  is the linear function and  $y$  is the prediction.
  - b) **Lasso Regression** the Least Absolute Shrinkage and Selection Operator (Lasso) [14] shrinks some independent features of the dataset and sets others to 0.
  - c) **Random Forest Regression** differs from the Random Forest Classification since the method predicts a number by calculating the average results of each tree.
  - d) **XGBoost** Extreme Gradient Boosting is a tree ensemble learning method similar to Random Forest. XGBoost uses the weak classifiers, trees which might have only two leaves. Upon the completion of training a tree ensemble, a new classifier is added to improve the trained tree.

### 3.2.3 Clustering methods

Clustering is a learning technique, where a structure has to be applied on an unlabeled dataset. The main purpose of clustering is to divide a dataset into groups (clusters), in a way that the elements within a cluster are more similar to each other than elements belonging to different clusters. There are many clustering algorithms proposed in the literature, hierarchical clustering, DBSCAN, k-means and k-medoids. In this subsection we will focus on the clustering algorithm which has been used in this project **k-means**.

**k-means:** Is a popular clustering method and one of the most used method. The method takes a numeric dataset  $X$  and an integer number  $k$  ( $n$ ), the algorithm aims to group the objects of  $X$  into  $k$  clusters in a way to minimise the sum of squared errors between the elements within a cluster [16].

### 3.3 A Web-Based Tool For Predictive Process Analytics

The tool developed in this project is an extension of the tool presented in [12] called Nirdizati Training.

#### 3.3.1 General Framework

The tool, consists of five main modules besides a storage module. The first module is the Front-end. The front-end allows the user to select settings used for the prediction and to visualize the prediction results. The second module is the Log Manager which is the module which handles uploading and retrieving the logs. The training phase is done in the third module, the Prediction Module. This module deals with the encoded log from the Encoder (the fourth module) which uses the logs from the Storage Module. Once the encoded log is loaded, the data is split into a training set which will be used to build the predictive model, and a test set which will be used for evaluating the created model. The last module is the Evaluation Module. This module aggregate the results of the test set and calculate the error of the created model [12]. Figure 6 explains the relations between the presented modules.

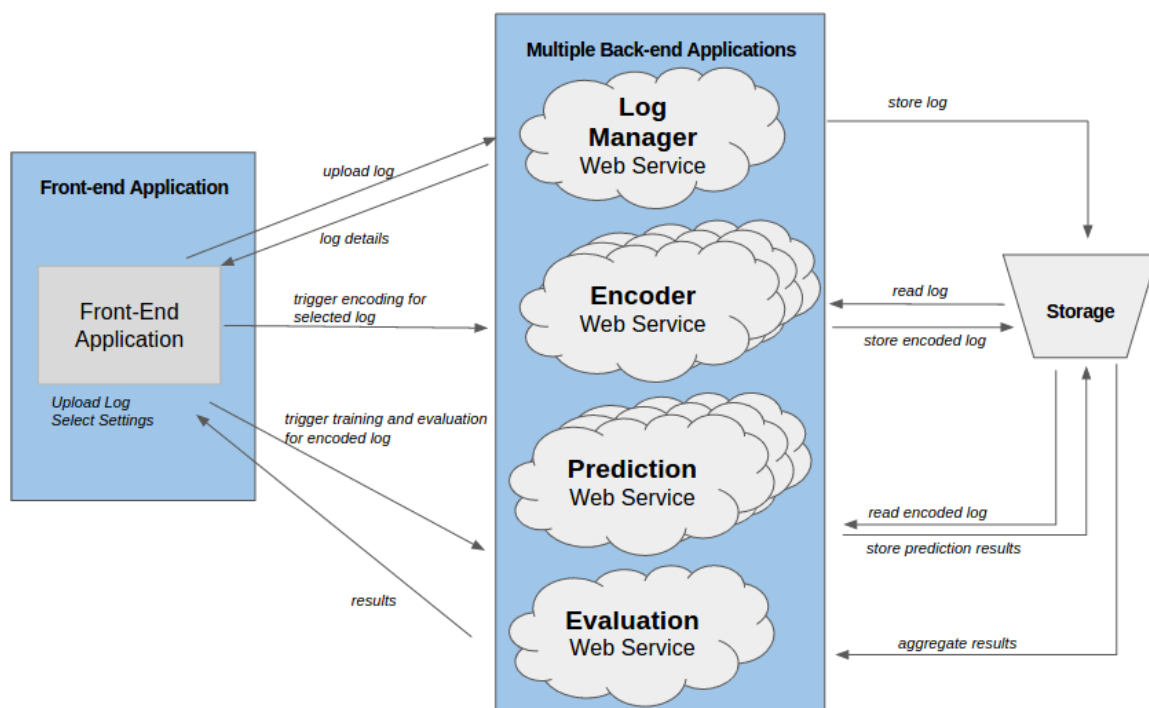


Figure 6. General Framework of a Nirdizati Training [12]

### 3.3.2 Back-end Application

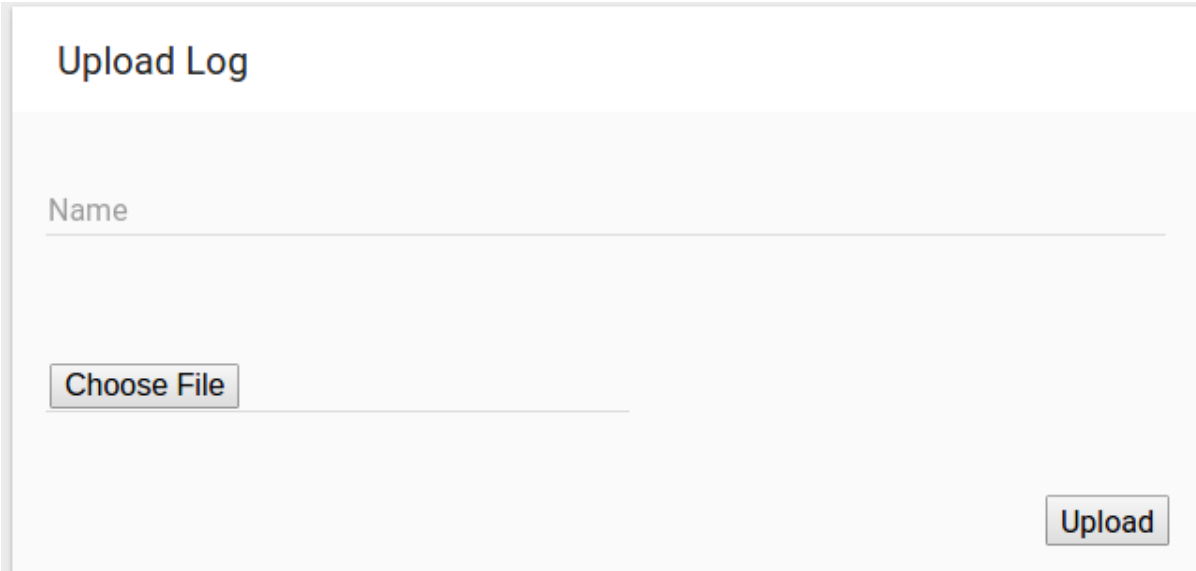
The back-end application was divided into two servers. Each server handles and provide web services: Django Back-end<sup>1</sup> and Java Spring Back-end<sup>2</sup> [12]. The Django Back-end have the following web services:

1. Log Manager: The log manager was the service which interfaces with uploading the log, querying the log to the encoder, and fetches the encoded log to the predictor.
2. Encoding: The service supported index-based encoding combined with different labels such as remaining time, next activity and conformance to a business rule.
3. Forecasting: The supported forecasting method is ARMA.
4. Regression: This service supported prediction methods, the supported methods are linear, lasso, random forest and xgboost regression algorithms.
5. Inter-case Prediction Service: This prediction method was implemented by taking into consideration intercase features shared by all cases of a log.

The Java Back-end supported the prediction methods for classifications, decision trees using the REPTree algorithm, Random Forest Classification and K-Nearest Neighbors (KNN).

### 3.3.3 Front-end Application

The front-end application was developed using AngularJS Material<sup>3</sup>. The front end application covered three main sections. These are the dashboard, prediction and log uploading and selection. The log uploading is shown in Figure 7, it allows a user to upload a log for analysis.



The screenshot shows a web form titled "Upload Log". It contains a text input field for "Name", a "Choose File" button for selecting a log file, and an "Upload" button at the bottom right.

Figure 7. Log Upload Page

---

1 <https://www.djangoproject.com/>

2 <https://projects.spring.io/spring-framework/>

3 <https://material.angularjs.org/latest/>

The prediction page provides the user three menus to select from, the log, the type of prediction and the method. The selection panel is shown in Figure 8.

The screenshot shows a web interface titled "Prediction". It contains three dropdown menus for selection: "Select Log: Select Log", "Select Type of Prediction: Select Type", and "Select Method of Prediction: Select Prediction Method". Below these menus is a "Submit" button.

Figure 8. Prediction Page

The predictions types and methods served by the selection panel are listed in Figure 9.

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

Figure 9. Prediction Types and Methods

## 4 Contribution

### 4.1 Framework for The Comparison of Predictive Process Monitoring Algorithms

In order to develop a tool for the Comparison of Predictive Process Monitoring Algorithms, we have updated the structure of the framework shown in Figure 6. We have included a queuing system which can be considered as the main interface of the tool. The front-end application allows the user to upload the log and configure the settings of the prediction algorithms to be compared. The log manager is the module responsible of uploading and retrieving the log. The Encoder job is to request the log from the log manager and encode the log to be feed to the Prediction module, which is responsible of creating the predictive model and provide the predictions. The evaluation module job is to aggregate, prepare and store the results in the storage module. The newly updated module is the queuing system. The queuing system job is to receive the configurations from the front-end, and assign each task to a worker which triggers the encoder and the prediction module. Figure 10 shows the new architecture of the framework.

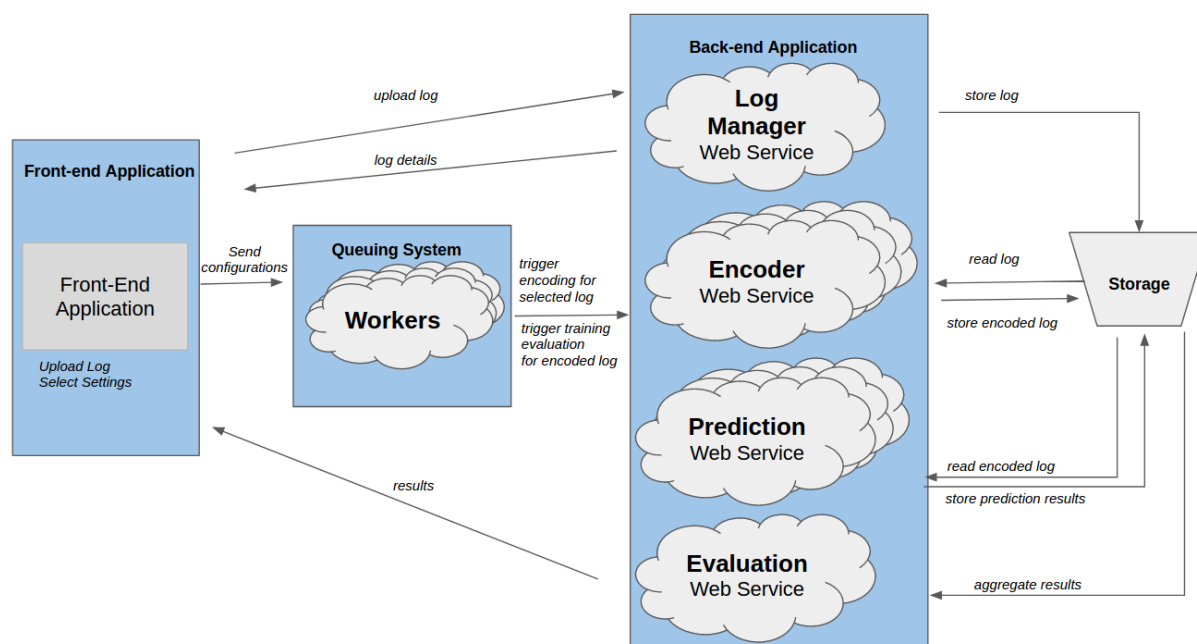


Figure 10. Framework for The Comparison of Predictive Process Monitoring Algorithms

#### 4.1.1 The Implemented Methods

- **Encoding:** Simple Index-Based Encoding, Boolean Encoding, Frequency Encoding, Index-Based Encoding and Latest Payload Encoding. Subsection 4.3 describes the encoding methods in detail.

- **Labelling:** For each trace we have included a label which will be predicted. For regression the label is Remaining Time, for classification Fast/Slow Remaining Time or Fast/Slow Duration. Remaining Time:  $RemainingTime = T_{en} - T_{ep}$  where  $T_{en}$  is the timestamp of the last event in the trace and  $T_{ep}$  is the timestamp of the event at the position of the prefix length. Duration:  $Duration = T_{en} - T_{e1}$  where  $T_{en}$  is the timestamp of the last event in the trace and  $T_{e1}$  is the timestamp of the first event in the trace. For classification calculate the label comparing the remaining time/duration of the current case with the threshold that can be user-specified or automatically computed as the average remaining time/duration of the cases in the input log.
- **Clustering:** K-means.
- **Classification:** K-Nearest Neighbor (KNN), Decision Trees and Random Forest.
- **Regression:** Linear Regression, Lasso Regression, Random Forest Regression and XGBoost.
- **Metrics:** For Classification predictions we have calculated Accuracy, Area Under The Curve, and F1 score, on the other hand for Regression predictions we have calculated Root Mean Square Error, Mean Absolute Error and Coefficient of Determination.

## 4.2 Types of Encoding

Encoding is an important stage for creating a predictive model. Choosing the right encoding for the dataset can be a difficult task. Therefore in our tool we have implemented all the proposed encoding methods in subsection 3.2.1. For us to minimise the execution and improve the performance of the tool we have merged the implementation of the five encoding methods into one function which outputs five different encoded logs from the training set. The function first starts by removing traces with size lower than the prefix length, once that done we loop through each trace. For **Simple Index-Based Encoding**, we recorded the sequence of the events. For **Boolean Encoding** and **Frequency Encoding**, we started the process with creating a list of all the unique events occurred in the log, and then for each trace, for boolean for each unique event in the list we check if it did appear in the current trace, if yes then we assign 1 otherwise 0. On the other hand, for frequency we followed the same process but instead of checking if an event occurred in a trace we count how many times it occurred and assign that number to the event. In the **Index-Based Encoding**, the sequence of the events and the event attributes are represented in the encoded trace. Finally for Index **Latest Payload Encoding**, we have represented the encoded trace as a sequence of events plus to the attributes of the latest event in the trace.

## 4.3 Clustering

In order for us to combine a k-means clustering into the creation of a predictive model we have implemented the procedure shown in Figure 11. Once the log has been retrieved and encoded, we split the data into two sets: the training set which is 80% of the whole log and the testing set which includes the remaining traces of the log. The training set is given as input to the k-means clustering in order to create a clustering model. As a result we get N training set clusters where each cluster has similar traces from the training set. Each cluster is used to build a classifier/regressor. Each prefix in the testing set is used for making predictions. The prefix is associated to a cluster and the corresponding regressor/classifier is used to get predictions on that prefix. The evaluation part is done by comparing the predictions with the ground truth derived from the completed traces in the testing set.



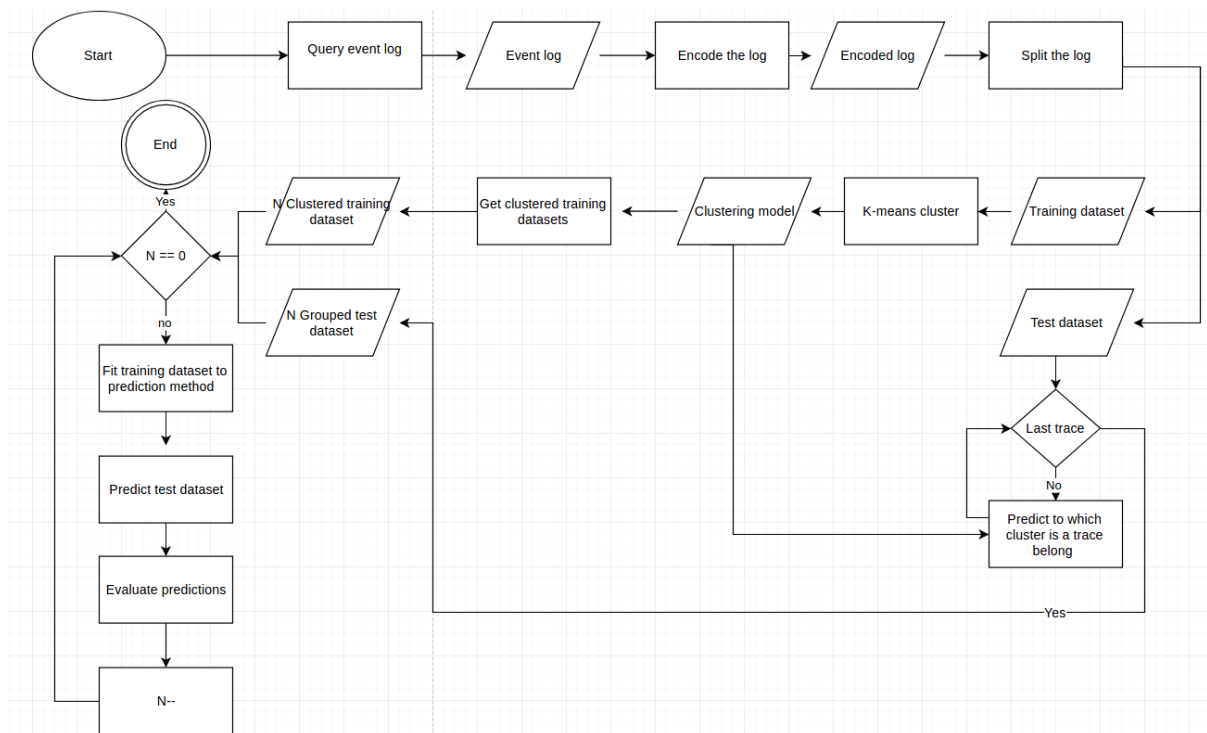


Figure 11. Clustering

## 5 Tool Implementation

In this section we introduce how the tool was implemented and the features available in the tool. We describe the back-end and the views in the front-end application.

### 5.1 Front-end Application

In this project we have extended the proposed front-end in [12]. The structure remains the same, we added views to interface the Queuing System and visualize the results coming from the Evaluation module.

#### 5.1.1 Views in the Application

The user first lands on the dashboard page. The page shows the details of the log. It shows the number of the active traces and resources per day (Figure 13). It also has details about the events in the log, in terms of the number of executions (Figure 12).

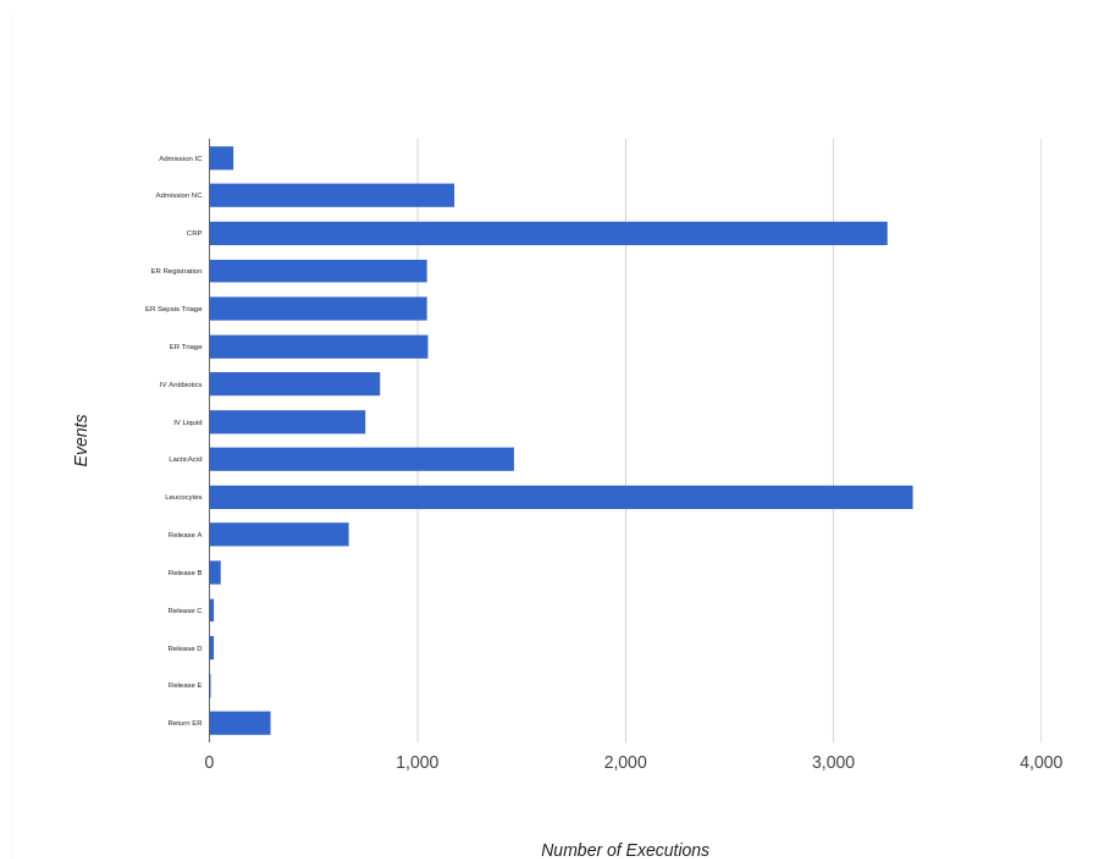


Figure 12. Number of Executions

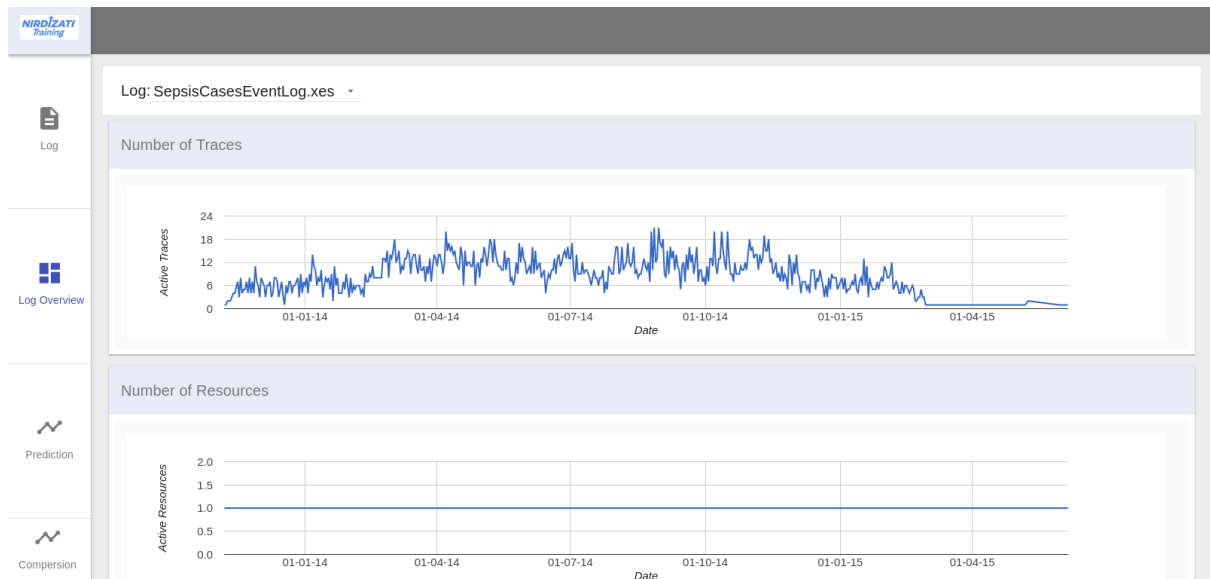


Figure 13. Log Dashboard

From the menu side bar the users will be able to navigate through the application. In this subsection we will focus on the *Comparison* tab. The table includes a menu to *Classification Configuration*, *Classification Results*, *Regression Configuration*, and *Regression Results* as shown in Figure 14.

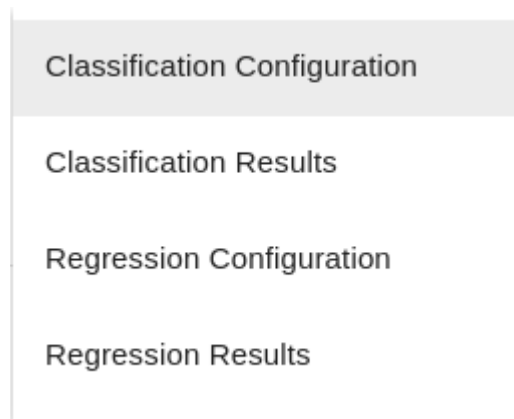


Figure 14. Comparison Menu

*Classification Configuration* page allows the user to select 30 combinations of different approaches to predict a business rule. The logic behind this control panel, is to help the user to run different approaches. The control panel includes different encoding, prediction methods, and it allows the user to combine the creation of the predictive model with a clustering method or not. The user needs to select a log. Since the predictions are done on a prefix trace the user has to enter the prefix length. The current available rules are: Fast/Slow based on the remaining time or Fast/Slow based on the trace duration. The user can enter a threshold to discriminate between fast and slow traces. The system can automatically set it to the average duration/ remaining time in the log. A screenshot of the control panel is shown in Figure 15.

Figure 15. Classification Configuration

*Regression Configuration page* is similar to the *Classification Configuration page*. It differs from it since in Regression we predict the remaining time of the traces. The user can choose different combination of encoding, prediction methods and the option to combine them with clustering method or not. The log should be selected, and the prefix length has to be entered. From this panel a user can run 40 different combination of regression predictions. A screenshot is shown in Figure 16.

Figure 16. Regression Configuration

*Classification Results page.* is the page where the user can check the results of the classification algorithms. The page lists the available results for classification predictions. From the page the user will be able to check which logs, prefix, rules, threshold, and which combinations results are available. The page have four main tabs. These tabs are General, Bubble chart by classifier, Bubble chart by clustering, Bubble chart by encoding. These tabs will be covered in subsection 6.2. The page includes a list of the available combinations as a check box. Once the user selects any, a new tab will be added which holds the full results of the selected combination. Figure 17 is a screenshot of the page.

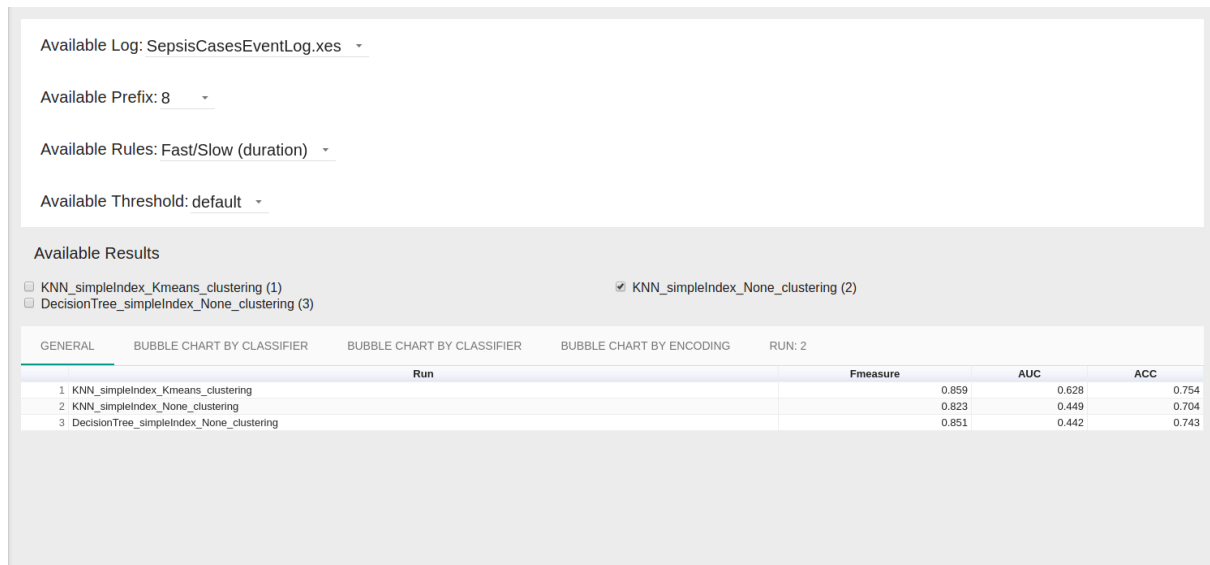


Figure 17. Classification Results

*Regression Results page*, lists the available regression combination of remaining time predictions. Similar to the classification results page it has four main tabs. These tabs are General, Bubble chart by classifier, Bubble chart by clustering, Bubble chart by encoding. These tabs will be covered in subsection 6.2. The page includes a list of the available combinations as a check box. Once the user selects any, a new tab will be added which holds the full results of the selected combination. Figure 18 is a screenshot of the page.

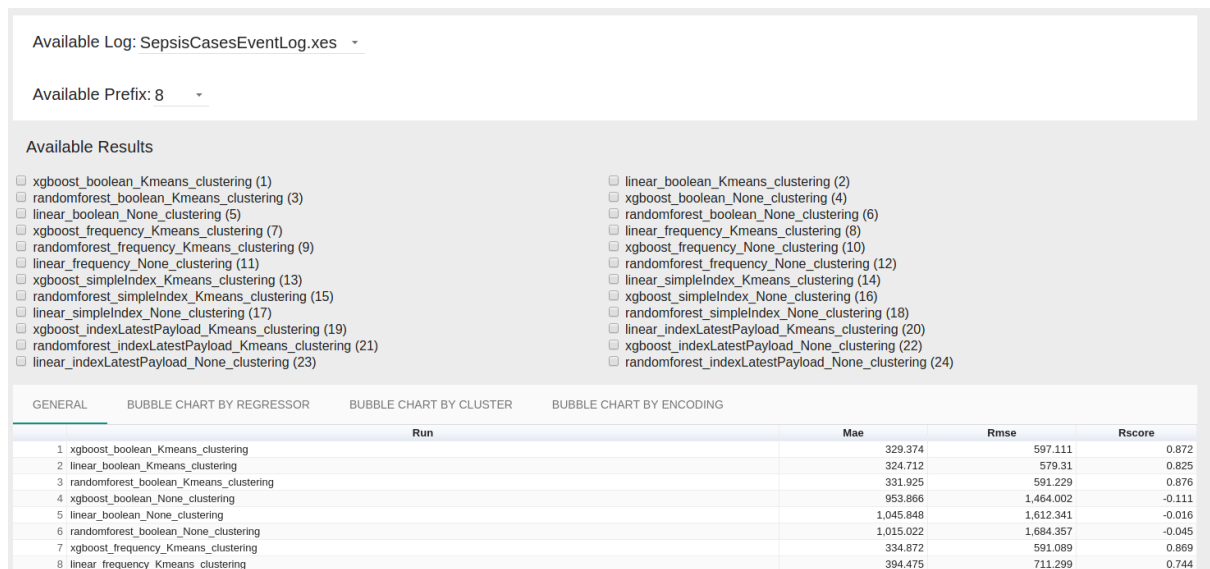


Figure 18. Regression Results Page

## 5.2 Back-end Application

The back-end application is based on the following web services:

- Log Manager - This service has been implemented by [12] and described in Section 3.3.2.1
- Encoding - To achieve the aim of providing The Comparison Of Predictive Process Monitoring Algorithms, this service supports multiple encoding methods. These methods are frequency encoding, boolean encoding, simple index encoding, simple index encoding with latest payload and complex index encoding.
- Clustering - After the encoding, this service provides the user with the ability of clustering the encoded logs with k-means clustering, or running the predictions directly on the encoded log.
- Regression - Our project has not added any new regression methods. It does support the methods mentioned in Section 3.3.2.1.
- Classification - In [12] the authors decided on splitting the back-end into two servers. Due to the nature of the back-end architecture shown in Figure 10, we have integrated the mentioned classification methods in Section 3.3.2.1 to our back-end.
- Evaluation - In this project we had to modify the presented Evaluation module in [12], to be front-end individual. The Evaluation module represent the results of each combination as a JSON array, and provides a general aggregated results for each configuration run. The general results and the individual predictions results can be requested by any front-end as long as they have the back-end endpoints.

## 5.3 Queuing System

The queuing system is the real interface of the back-end. The aim of the queuing system is to allow the user to run multiple combination of different encoding methods with different prediction methods. Currently the system has two end points to receive the user's settings, *Classification Configuration* and *Regression Configuration*. The end points are developed to be front-end individual. By front-end individual, we mean that the mentioned end points receives the information through a HTTP post request with JSON<sup>4</sup> data which allows future development to use the same back-end with a different front end. After the user sends the post request the queuing system assigns tasks to the running workers, and the system can have n workers. Each worker will handle one task. After the completion of that task the system will assign a new task to the same worker till the queued tasks are all completed. In this project we used Django-RQ<sup>5</sup> queuing system, which is a simple application which allows us to queue tasks in our Django Server. The application provides an admin section which let the user check the queuing pipelines, how many tasks are queued, how many have been completed, how many has failed and how many workers are running. The default application settings has three executions priorities: high, low and default. For simplicity we only assigned tasks to the default queuing pipeline. A screenshot of how the RQ Queues table looks like is shown in Figure 19.

---

4 <http://www.json.org/>

5 <https://github.com/ui/django-rq>

## RQ Queues

NAME	QUEUED JOBS	ACTIVE JOBS	DEFERRED JOBS	FINISHED JOBS	NUMBER OF WORKERS	HOST	PORT	DB
default	0	0	0	0	0	0.0.0.0	6379	0
high	0	0	0	0	0	0.0.0.0	6379	0
low	0	0	0	0	0	0.0.0.0	6379	0
failed	0	-	-	-	-	0.0.0.0	6379	0

*Figure 19. RQ Queues*

### 5.3.1 Classification Configuration

The classification configuration allows the user to choose one of five different encoding methods listed in subsection 3.2.1 and three classification prediction method listed in subsection 3.3.2.1. For this type of predictions we proposed two different rules, fast/slow prediction based on the remaining time, or fast/slow prediction based on the duration of a trace. The architecture of this configuration allows the user to enter a threshold or choose the “default” threshold. This means that we calculate the average of the selected rule and use this value as threshold for us to categorize slow/fast traces of the training dataset. Also the user can choose to combine the prediction with k-means clustering or without any clustering (for that option the keyword “None” is used). The predictions are done on prefix traces, therefore the prefix length has to be included in the JSON along with the name of the log. The user can queue a combination of 30 prediction tasks with a single JSON post. An example is provided in Figure 20.

```
{
  "log": "SepsisCasesEventLog.xes",
  "prefix": 7,
  "encoding": ["complexIndex", "boolean", "frequency", "indexLatestPayload", "simpleIndex"],
  "clustering": ["Kmeans", "None"],
  "classification": ["KNN", "RandomForest", "DecisionTree"],
  "rule": "remainingTime",
  "threshold": "default"
}
```

*Figure 20. Classification Configuration JSON*

### 5.3.2 Regression Configuration

This configuration is similar to the Classification Configuration. It differs since in Regression we are predicting the remaining time then we do not need a threshold value. The user is still allowed to queue at maximum 40 different combinations of remaining time predictions using this configuration by making HTTP post request with JSON data defining which encoding methods, prediction methods, predictions with or without clustering and prefix along with the name of the log. A JSON example is provided in Figure 21.

```
"log": "SepsisCasesEventLog.xes",
"prefix": 8,
"encoding": ["simpleIndex", "boolean", "frequency", "indexLatestPayload", "complexIndex"],
"clustering": ["Kmeans", "None"],
"regression": ["xgboost", "linear", "randomforest", "lasso"]
```

Figure 21. Regression Configuration JSON



## 6 Evaluation and Verification

### 6.1 Evaluation Measures

To provide the researchers with a tool to compare different prediction approaches we have evaluated the created predictive models by using some well known predictions measures. For Regression predictions we have calculated **Root Mean Square Error**, **Mean Absolute Error**, and **Coefficient Of Determination**. For the classification predictions we have calculated **Accuracy**, **Area Under The Curve**, and **F1 Score**.

#### 6.1.1 Root Mean Square Error

Root Mean Square Error (RMSE) calculates sample standard deviation on the differences between the actual and the predicted values. RMSE can be used when we are predicting numerical values and it can be calculated with the following formula:

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

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

#### 6.1.2 Mean Absolute Error

Mean Absolute Error calculate the average vertical distances between the actual and the predicted values. To calculate the distance between the observed points the values have to be numerical and it can be calculated with the following formula:

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

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

#### 6.1.3 Coefficient Of Determination

Coefficient Of Determination or “R squared” is the square of the measure of the linear correlation between the actual and the predicted values. It can be calculated with the following formula:

$$r^2 = \left( \frac{n(\sum xy) - (\sum x)(\sum y)}{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]} \right)^2$$

Where  $n$  is the number of rows in the dataset,  $x$  is the predicted and  $y$  is the actual value.

#### 6.1.4 Accuracy

Accuracy it is a measure of statistical bias, a description of systematic errors. Accuracy can only be used on binary classification, the formula is the following:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP (True Positive) is the number of positive predictions correctly classified, TN (True Negative) is the number of negative predictions correctly classified, FP (False Positive) is the number of negative predictions classified as positive and FN (False Negative) is the number of positive predictions classified as negative. These values can be calculated from the confusion matrix. Figure 22 is a representation of a confusion matrix.

	<b>True (predicted)</b>	<b>False (predicted)</b>
<b>True (actual)</b>	TP	FN
<b>False (actual)</b>	FP	TN

Figure 22. Confusion Matrix

### 6.1.5 Area Under The Curve

For Area Under The Curve, we compute the area under the ROC curve, ROC stands for Receiver Operating Characteristic. In this project we used the python function provided by sklearn.metrics<sup>6</sup>. The function takes a vector of probabilities and vector of the actual values.

### 6.1.6 F1 Score

F1 Score is a weighted average of the precision and recall, F1 Score is a value between 1 and 0, the closer to 1 is the better score. The formula is:

$$F1 = \frac{2TP}{2TP + FP + FN}$$

## 6.2 Results

In this section we provide the comparison results provided by the implemented tool using a real-life log pertaining to the treatment process of sepsis patients in a hospital. The log includes data from Nov 7, 2013 till June 5, 2015. It contains 4531 traces, 15214 activities distributed among 16 unique activities. This section has two subsection: Classification Results and Regression Results. The results shown in the subsections have a prefix length equals to eight.

### 6.2.1 Regression Results

The results are represented in three different manners: row aggregated results (General), three different bubble charts and trace per trace results.

**General** This tab includes the results of all the different regression combination related to a log and a prefix. Each row in the table represents a single combination and shows measurement scores as MAE, RMSE, and R squared. An example of the results can be found in Figure 23.

---

6 <http://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html>

GENERAL	BUBBLE CHART BY REGRESSOR	BUBBLE CHART BY CLUSTERING	BUBBLE CHART BY ENCODING	
Run	Mae	Rmse	Rscore	
1	lasso_boolean_Kmeans_clustering (28)	297.021	471.251	0.898
2	linear_frequency_Kmeans_clustering (11)	302.4	466.254	0.91
3	xgboost_boolean_Kmeans_clustering (25)	304.75	543.504	0.834
4	linear_simpleIndex_Kmeans_clustering (3)	309.716	580.362	0.875
5	lasso_simpleIndex_Kmeans_clustering (4)	312.469	545.724	0.89
6	xgboost_frequency_Kmeans_clustering (9)	315.104	541.071	0.858
7	linear_boolean_Kmeans_clustering (27)	322.054	506.464	0.846
8	randomforest_simpleIndex_Kmeans_clustering (2)	325.66	505.683	0.86
9	randomforest_indexLatestPayload_Kmeans_clustering (18)	327.607	516.942	0.862
10	randomforest_frequency_Kmeans_clustering (10)	343.201	662.349	0.86
11	linear_indexLatestPayload_Kmeans_clustering (19)	345.94	628.896	0.836
12	randomforest_boolean_Kmeans_clustering (26)	349.302	602.342	0.829
13	xgboost_indexLatestPayload_Kmeans_clustering (17)	357.29	576.209	0.822
14	xgboost_simpleIndex_Kmeans_clustering (1)	360.008	677.218	0.756
15	xgboost_complexIndex_Kmeans_clustering (33)	365.381	553.993	0.857
16	lasso_frequency_Kmeans_clustering (12)	367.201	770.416	0.82
17	randomforest_complexIndex_Kmeans_clustering (34)	424.706	671.565	0.766
18	lasso_indexLatestPayload_Kmeans_clustering (20)	659.972	3,614.84	-5.554
19	linear_complexIndex_None_clustering (39)	849.496	1,210.439	-0.079
20	linear_simpleIndex_None_clustering (7)	896.532	1,344.95	-0.041
21	linear_indexLatestPayload_None_clustering (23)	898.366	1,413.996	-0.004
22	randomforest_frequency_None_clustering (14)	912.433	1,556.211	0.006
23	xgboost_frequency_None_clustering (13)	916.212	1,312.499	-0.173
24	lasso_simpleIndex_None_clustering (8)	925.224	1,378.134	-0.008
25	lasso_frequency_None_clustering (16)	929.961	1,485.909	-0.016
26	lasso_boolean_None_clustering (32)	943.374	1,468.332	-0.018
27	xgboost_boolean_None_clustering (29)	943.936	1,456.083	-0.023
28	linear_frequency_None_clustering (15)	955.025	1,570.956	-0.016
29	randomforest_boolean_None_clustering (30)	985.538	1,515.855	-0.113
30	randomforest_complexIndex_None_clustering (38)	986.488	1,591.101	-0.24

Figure 23. General Regression Tab

The results table can be sorted based on each metric. From Figure 23, we can see that for the selected log and prefix a Boolean Encoding combined with K-means clustering predicted by Lasso method (28) have the lowest MAE value. Frequency Encoding with K-means clustering predicted by Linear Regression (11) has the second lowest MAE value and the lowest RMSE value. After we sorted the table based on R squared score we were able to see that the combination of complex index encoding, linear regression with k-means clustering has scored the worst scores.

**Bubble Charts** We used MAE as x-axis and RMSE as y-axis, the size of the bubble is R squared, and the bubble Id is the unique Id of a combination. A bubble is shown in Figure 24.

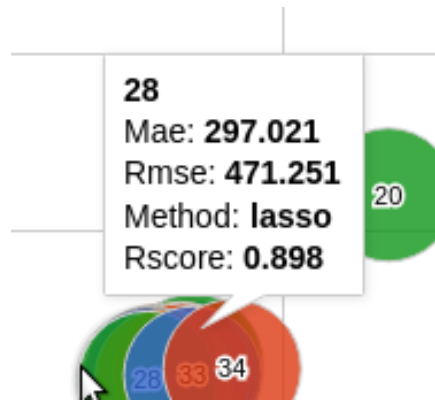


Figure 24. Regression Bubble

For better comparison three different bubble charts are provided, all the three follow the same concept and they differ by the bubble colour. In the *BY REGRESSOR* chart the different colours represent the regression method. An example can be seen in Figure 25.

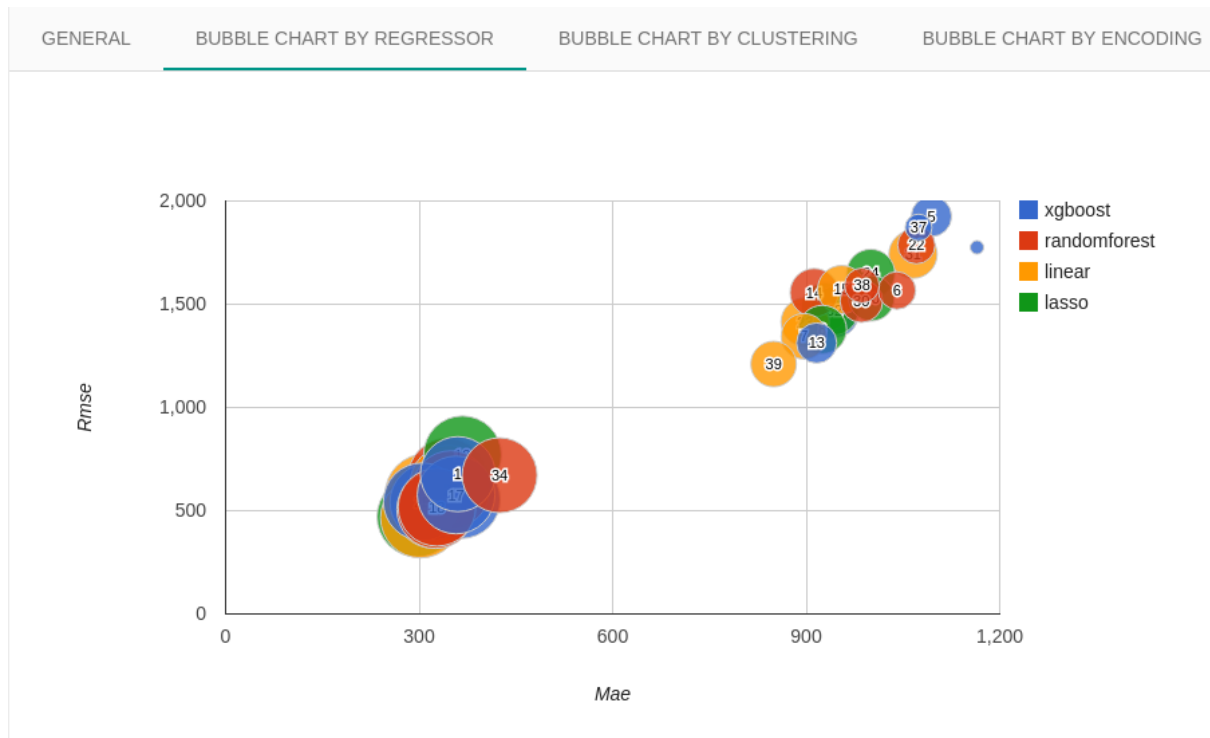


Figure 25. Regression By Regressor bubble chart

From Figure 25, we can see that all the regression methods used have low MAE and RMSE. The group of bubbles with MAE less than five hundred and RMSE less than eight hundred can be considered as the best combinations for the mentioned log with a prefix length equal to eight. The combination of Index Latest Encoding with XGBoost regression (21) has the highest MAE and the lowest R squared score. The combination of Simple Index Encoding with XGBoost regression (5) has the highest RMSE.

In the *BY CLUSTERING* chart the colours represent the clustering methods. In this project we have only two possibilities: with k-means clustering or without any clustering. An example can be seen in Figure 26.

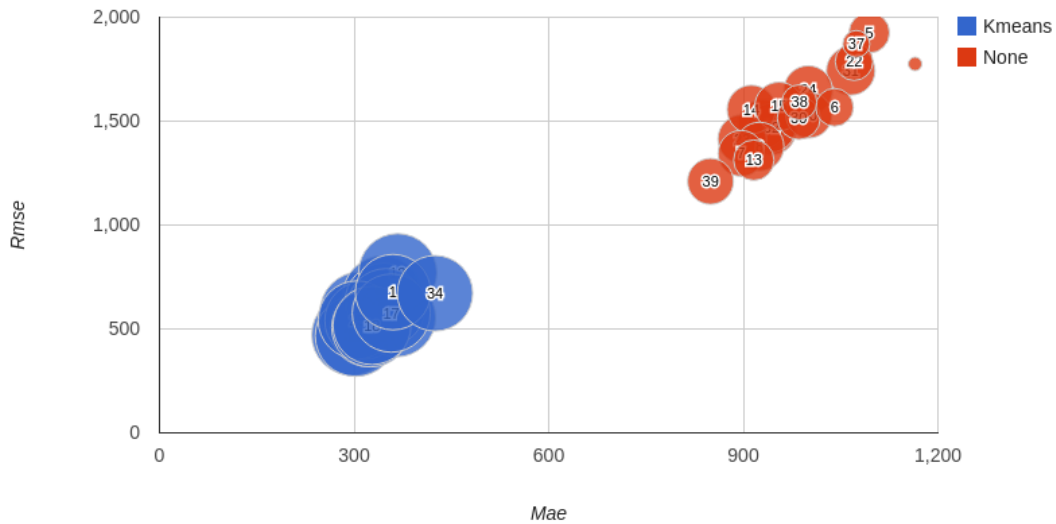


Figure 26. Regression By Clustering bubble chart

In Figure 26, we can see that clustering improves the predictions since most of the blue bubbles (K-means clustered) have lower MAE, RMSE than the red ones which have not been combined with any clustering.

Finally in the *BY ENCODING* chart the colours represent the used encoding method. An example can be seen in Figure 27.

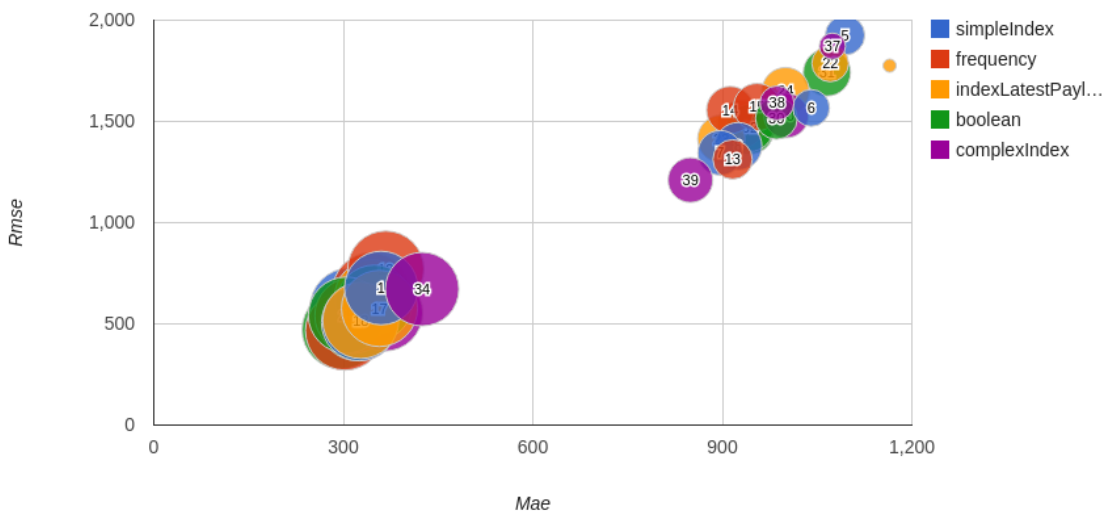


Figure 27. Regression By Encoding bubble chart

**Trace Per Trace** in this table we show the row data of the actual traces. In particular we list the traces ids and their actual and predicted Remaining time . An example can be seen in Figure 28.

GENERAL	BUBBLE CHART BY REGRESSOR	BUBBLE CHART BY CLUSTERING	BUBBLE CHART BY ENCODING	RUN: 28
<b>Trace Id</b>	<b>Actual RemainingTime</b>	<b>Prediction</b>		
V	7123931.0	9631851.65488		
KB	12479692.0	9631851.65488		
ED	14469757.0	9631851.65488		
FJ	10631445.0	9631851.65488		
HL	8583109.0	9631851.65488		
EO	15916356.0	9631851.65488		
QO	10278134.0	9631851.65488		
UQ	10124943.0	10600981.9191		
XQ	6559031.0	9631851.65488		
OS	8893231.0	9631851.65488		
IT	8739831.0	9631851.65488		
UV	6748065.0	9631851.65488		
BX	8675630.0	9631851.65488		
XY	15015623.0	8988097.09651		
WZ	5421564.0	9631851.65488		
YZ	11748181.0	9631851.65488		
YDA	6628938.0	9631851.65488		
AFA	7532304.0	9025690.01412		
PIA	12489005.0	9631851.65488		
A	954673.0	851753.891829		
B	451565.0	851753.891829		
G	585540.0	851753.891829		
T	191460.0	851753.891829		
FA	0.0	851753.891829		

Figure 28. Regression Per Trace Results

## 6.2.2 Classification Results

As in Regression Results the classification results are represented in three different manners: row aggregated results (General), three different bubble charts and trace per trace results.

**General** This tab includes the results of all the different classification combination related to a log, a prefix and a rule. Each row in the table represents a single combination and measurement scores as AUC, F1 Score, and ACC. An example of the results can be found in Figure 29.

GENERAL	BUBBLE CHART BY CLASSIFIER	BUBBLE CHART BY CLUSTERING	BUBBLE CHART BY ENCODING			
Run				Fmeasure	AUC	ACC
1	KNN_frequency_Kmeans_clustering (8)			0.969	0.466	0.95
2	RandomForest_frequency_Kmeans_clustering (9)			0.969	0.375	0.95
3	RandomForest_boolean_Kmeans_clustering (21)			0.965	0.569	0.944
4	DecisionTree_boolean_Kmeans_clustering (19)			0.965	0.532	0.944
5	KNN_boolean_Kmeans_clustering (20)			0.965	0.425	0.944
6	DecisionTree_frequency_Kmeans_clustering (7)			0.965	0.354	0.944
7	DecisionTree_boolean_None_clustering (22)			0.873	0.503	0.777
8	RandomForest_frequency_None_clustering (12)			0.873	0.438	0.777
9	RandomForest_boolean_None_clustering (24)			0.869	0.51	0.771
10	DecisionTree_frequency_None_clustering (10)			0.869	0.428	0.771
11	KNN_boolean_None_clustering (23)			0.862	0.519	0.76
12	KNN_frequency_None_clustering (11)			0.855	0.463	0.749
13	KNN_simpleIndex_None_clustering (5)			0.853	0.417	0.749
14	KNN_indexLatestPayload_None_clustering (17)			0.848	0.454	0.737
15	KNN_complexIndex_None_clustering (29)			0.844	0.504	0.732
16	KNN_indexLatestPayload_Kmeans_clustering (14)			0.844	0.454	0.732
17	KNN_complexIndex_Kmeans_clustering (26)			0.838	0.501	0.721
18	RandomForest_simpleIndex_None_clustering (6)			0.833	0.471	0.721
19	RandomForest_complexIndex_None_clustering (30)			0.821	0.514	0.709
20	RandomForest_complexIndex_Kmeans_clustering (27)			0.808	0.511	0.687
21	DecisionTree_simpleIndex_None_clustering (4)			0.8	0.468	0.676
22	RandomForest_indexLatestPayload_Kmeans_clustering (15)			0.799	0.486	0.682
23	DecisionTree_indexLatestPayload_None_clustering (16)			0.781	0.505	0.659
24	RandomForest_indexLatestPayload_None_clustering (18)			0.774	0.456	0.637
25	DecisionTree_complexIndex_None_clustering (28)			0.758	0.465	0.626
26	DecisionTree_indexLatestPayload_Kmeans_clustering (13)			0.739	0.473	0.609
27	DecisionTree_complexIndex_Kmeans_clustering (25)			0.722	0.465	0.592
28	KNN_simpleIndex_Kmeans_clustering (2)			0.551	0.68	0.386
29	RandomForest_simpleIndex_Kmeans_clustering (3)			0.524	0.574	0.366
30	DecisionTree_simpleIndex_Kmeans_clustering (1)			0.503	0.444	0.346

Figure 29. General Classification tab

The results table can be sorted based on each metric. In Figure 29, we can see that for the selected log, rule and prefix the Frequency Encoding combined with K-means clustering predicted by K-Nearest Neighbor and Random Forest methods (8) and (9) have the highest F1 Score and ACC values. The run with Random Forest (9) has lower AUC value.

**Bubble Charts** We used F1 Score as x-axis and AUC as y-axis, the size of the bubble is ACC, and the bubble Id is the unique Id of a combination. A bubble is shown in Figure 30.

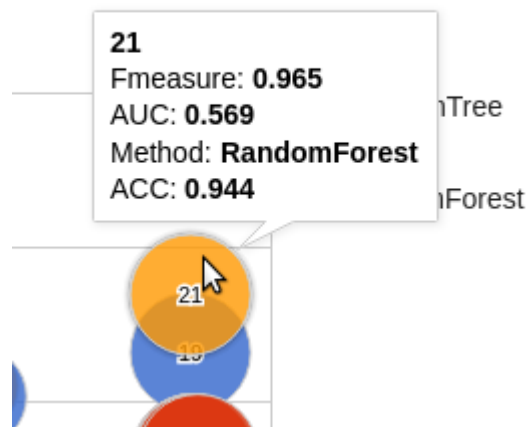


Figure 30. Classification Bubble

For better comparison three different bubble charts are provided, all the three follow the same concept and they differ by the bubble colour. In the *BY CLASSIFIER* chart the different colours represent the classification methods. An example can be seen in Figure 31.

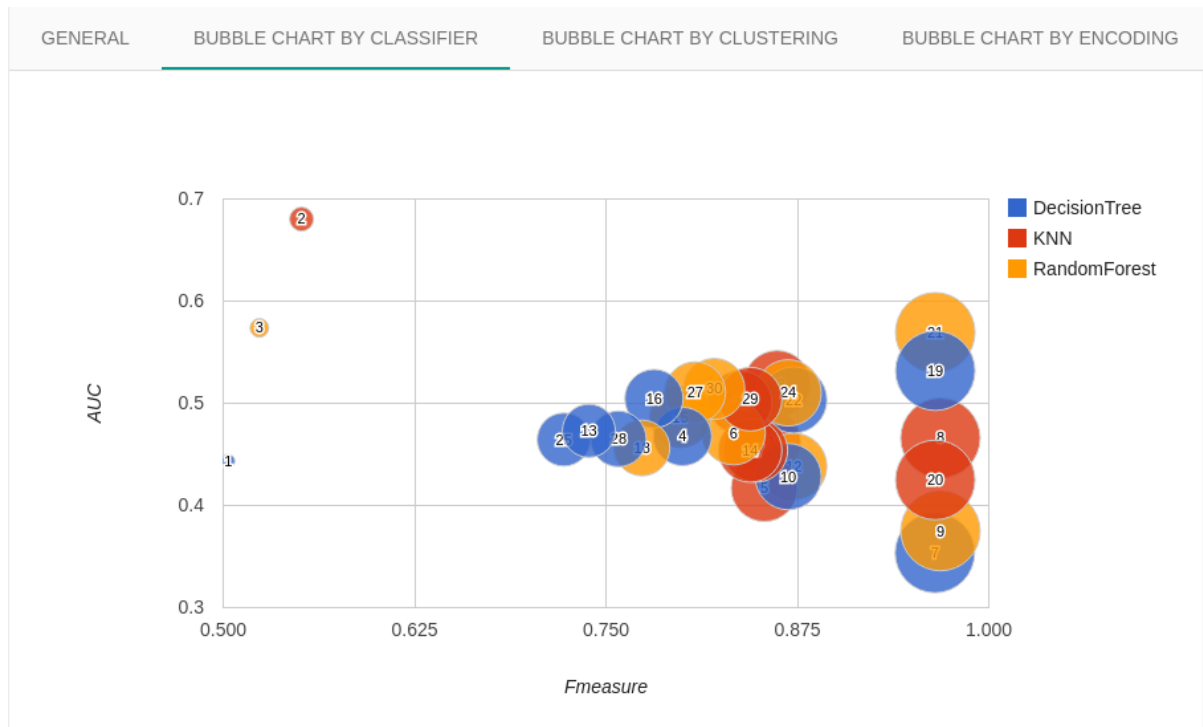


Figure 31. Classification By Classifier bubble chart

From Figure 31, we can see that The combination of Simple Index Encoding with K-Nearest Neighbor classifier combined with K-means clustering (2) has a low F1 score and the highest AUC. Simple Index Encoding with Decision tree classifier combined with K-means clustering (1) can be considered as the worst case since it has the lowest AUC and a low F1 score. Boolean Encoding with Random Forest classifier combined with K-means clustering (21) can be considered as the best fit for the selected log.

In the *BY CLUSTERING* chart the colours represent the clustering methods. An example can be seen in Figure 32.



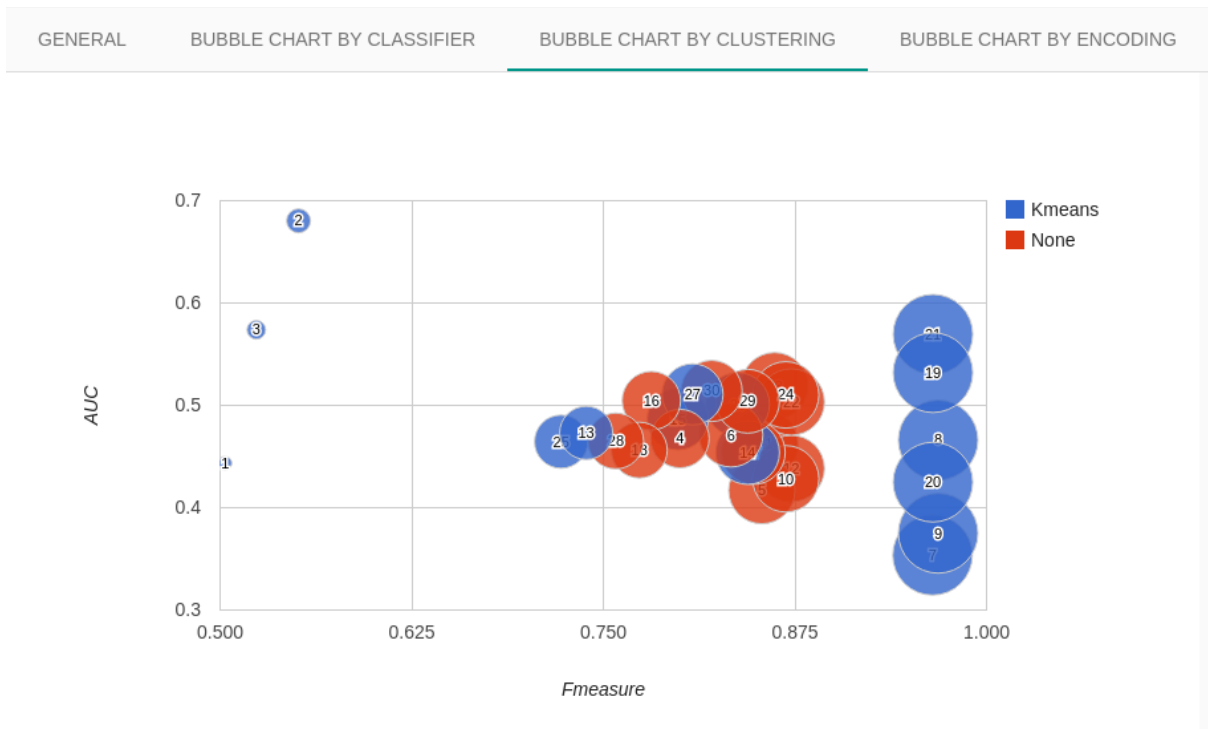


Figure 32. Classification By Cluster bubble chart

In Figure 32, as in regression we can see that clustering improves the predictions since most of the blue bubbles (K-means clustered) have high F1 Score, AUC and they are the biggest which means they have high ACC as well.

In the *BY ENCODING* chart the colours represent the used encoding method. An example can be seen in Figure 33.

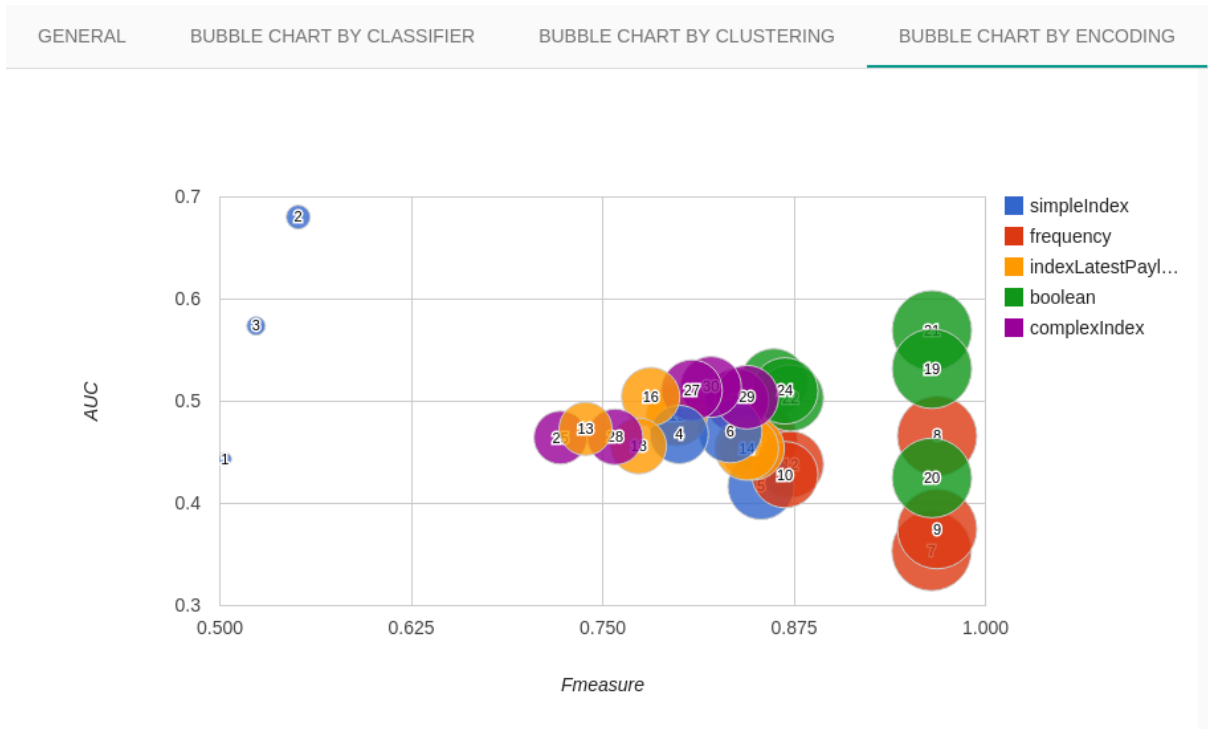


Figure 33. Classification By Encoding bubble chart

In Figure 33, we can see that Boolean encoding (the green bubbles) has better results than other encoding methods, since two of the highest values are runs encoded with boolean encoding. In general we can see that the green bubbles are performing better than others. The Frequency encoding comes in the second place. Combining the information from Figure 32 and Figure 33 we can see that the combinations of Simple Index encoding combined with K-means clustering can be considered as the worst for this log.

**Trace Per Trace** in this table we show the row data of the actual traces, where we list the traces ids and their actual and predicted outcome. An example can be seen in Figure 34.

GENERAL	BUBBLE CHART BY CLASSIFIER	BUBBLE CHART BY CLUSTERING	BUBBLE CHART BY ENCODING	RUN: 24
<b>Trace Id Actual Prediction</b>				
PFA	Fast	Fast		
MFA	Slow	Fast		
MJ	Fast	Fast		
QJA	Fast	Fast		
NN	Fast	Fast		
TF	Fast	Fast		
GDA	Slow	Fast		
HLA	Fast	Fast		
GV	Fast	Fast		
XKA	Slow	Fast		
EDA	Fast	Fast		
HI	Fast	Slow		
BP	Fast	Fast		
HF	Fast	Fast		
JBA	Fast	Fast		
BAA	Fast	Fast		
ZGA	Slow	Fast		
VQ	Fast	Fast		
LZ	Slow	Slow		
DEA	Fast	Fast		
PMA	Fast	Fast		
XV	Fast	Fast		
AGA	Fast	Fast		
PEA	Fast	Fast		
LP	Fast	Fast		
KB	Slow	Fast		
ES	Fast	Fast		

*Figure 34: Classification Per Trace*

## 7 Conclusion

This thesis proposes a tool comparison of predictive process monitoring algorithms. The tool allows the user to create predictive models and compare different combinations of Regression prediction algorithms, and different combinations of Classification prediction algorithms. Regression methods predict the remaining time and classification methods mainly focus on predicting two possible rules: Fast/Slow based on the remaining time of a trace, and Fast/Slow based on the duration of a trace. We were able to see prediction results of 40 combinations of regression algorithms, and 30 combinations for each classification rule.

The tool supports the user in answering the following questions:

- What is the suitable Encoding method?
- Do clustering add any value to the predictive model?
- In case of remaining time predictions which is the most suitable regression method?
- Which classifier should be used in the outcome based predictions?

In general choosing the suitable prediction method strongly depends on the given log, and with the help with the proposed tool this complex consuming task became easier.

In the future, we would like to improve the tool for the existing predictions by extend the possible combinations in both regression and classification, by including more Encoding, Clustering, Regression and Classification methods, and including other predictions than remaining time and outcome based predictions. For Classification adding more possibilities for the business rules. One more improvement would be improving the tool to have an interface exploiting the created predictive models to provides run-time predictions. The implemented tool code is publicly available in Bitbucket repositories, front-end<sup>7</sup>, backend<sup>8</sup>

---

7 [https://bitbucket.org/aytaleb/thesis\\_frontend](https://bitbucket.org/aytaleb/thesis_frontend)

8 [https://bitbucket.org/aytaleb/thesis\\_backend](https://bitbucket.org/aytaleb/thesis_backend)

## **8 Acknowledgement**

I would like express my sincere gratitude to my family, friends for the support and the encouragement during my trip attaining a Master's degree.

I would like to express my appreciation to my supervisors Fabrizio Maria Maggi and Fredrik Payman Milani for their guidance and support, also big thanks to the academic staff of the Computer Science faculty at the University of Tartu.

Last but not least I would like to thank my colleagues at Click & Grow OÜ for the support and the fixable working hours.

## References

- [1] 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.
- [2] 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.
- [3] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. *Predictive Monitoring of Business Processes*, pages 457–472. Springer International Publishing, Cham, 2014.
- [4] 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.
- [5] 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.
- [6] W. van der Aalst, W. et al. Process mining manifesto. In Daniel, F., Barkaoui, K., & Dustdar, S. (Eds.), *Business Process Management Workshops, Lect Notes Bus Inf*, (Vol. 99 pp. 169–194). Berlin: Springer, 2012.
- [7] 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.
- [8] 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.
- [9] Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: *Proc. of ICSOC*. pp. 389–403. Springer (2013).
- [10] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. Time and activity sequence prediction of business process instances. *CoRR*, abs/1602.07566, 2016.
- [11] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In *Proceedings of the 29th 44 International Conference on Advanced Information Systems Engineering*, page To appear. Springer, 2017.

- [12] Kerwin Jorbina, Fabrizio Maria Maggi, Chiara Di Francescomarino and Chiara Ghidini. A Web-Based Tool For Predictive Process Analytics, 2017.
- [13] Breiman, Leo. Random forests. *Machine learning*, 45(1), pp.5-32, 2001.
- [14] Tibshirani, Robert. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*. pp. 267-288, 1996.
- [15] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
- [16] Huang, Zhexue. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery* 2, no. 3, pp: 283-304, 1998.
- [17] W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer Berlin Heidelberg, 2016.

## **Appendix**

### **I. License**

#### **Non-exclusive licence to reproduce thesis and make thesis public**

I, **Ayham Taleb**,

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 Tool For The Comparison Of Predictive Process Monitoring Algorithms,**

supervised by Fabrizio Maria Maggi and Fredrik Payman Milani.

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, **29.08.2017**