Expert Systems With Applications 129 (2019) 233–245

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An empirical comparison of classification techniques for next event prediction using business process event logs

Bayu Adhi Tama, Marco Comuzzi*

School of Management Engineering, Ulsan National Institute of Science and Technology, Republic of Korea

ARTICLE INFO

Article history:
Received 30 July 2018
Revised 8 April 2019
Accepted 8 April 2019

Keywords:
Process indicators
Classification algorithms
Significance test
Performance evaluation
Event log
Empirical benchmark

ABSTRACT

Predictive analytics is an essential capability in business process management to forecast future status and performance of business processes. In this paper, we focus on one particular predictive monitoring task that is solved using classification techniques, i.e. predicting the next event in a case. Several different classifiers have been recently employed in the literature in this task. However, a quantitative benchmark of different classifiers is currently lacking. In this paper, we build such a benchmark by taking into account 20 classifiers from five families, i.e. trees, Bayesian, rule-based, neural and meta classifiers. We employ six real-world process event logs and consider two different sampling approaches, i.e. case and event-based sampling, and three different validation methods in order to acquire a comprehensive evaluation about the classifiers' performance. According to our benchmark, the classifier most likely to be the overall superior performer is the credal decision tree (C-DT), followed by the other top-4 performers, i.e. random forest, decision tree, dagging ensemble, and nested dichotomies ensemble. We also provide a qualitative discussion of how features of an event log can affect the choice of best classifier.

1. Introduction

Process mining in the last 15 years has drawn a large amount of attention from researchers in the field of business process management (BPM). It deals with the extraction of meaningful and possibly previously unknown information from the historical event logs and other business process data (van der Aalst, 2016). Predictive monitoring of business processes is one branch of process mining that aims at predicting the future value of valuable information about a business process. This allows decision makers to determine actions pro-actively (Teinemaa, Dumas, Maggi, & Di Francescomarino, 2016). For instance, if a model that predicts the occurrence of future events in a process is available, an alert could be sent to decision makers when specific events, e.g., risky or dangerous, are likely to occur in the future.

In the problem of process predictive monitoring, the input is represented by a process event log, which contains information about process execution in the past. This may range from labels of executed activities and timestamps, to any other attribute of interest about the execution of a specific event, such as the resource that executed the event or its cost. The output of a process predictive model is a prediction of some aspect of interest for a process, which is obtained by applying the model to the new instances. Several different aspects can be predicted, either numerical, categorical, or boolean, such as completed timestamp of activities in a process (numeric), process outcomes, e.g., whether the value of a process performance indicator will be above threshold (normally boolean or categorical), or the next event to occur (categorical). Based on the nature of the prediction to be made, process predictive monitoring is reduced to either a classification or regression problem.

A plethora of techniques for business process predictive monitoring has been developed in the past five years. Márquez-Chamorro, Resinas, and Ruiz-Cortés (2017) have published an in-depth qualitative review of techniques and application scenarios in business process predictive monitoring. In this paper, rather than simply collecting and classifying the existing approaches by conducting an all-purpose literature survey, we focus on the quantitative evaluation of process predictive monitoring techniques. A practical problem for decision makers in process predictive monitoring, in fact, is often the one of choosing the best-performing technique in a given application scenario. This can be achieved only by building a quantitative comparison benchmark in which different techniques are evaluated for a particular application scenario in a variety of settings.

* Corresponding author.
E-mail addresses: bayu@unist.ac.kr (B.A. Tama), mcomuzzi@unist.ac.kr (M. Comuzzi).

https://doi.org/10.1016/j.eswa.2019.04.016
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

As an application scenario, this paper focuses on prediction of the next event in a case, which is one of the principal use cases in business process predictive monitoring (Márquez-Chamorro, Resinas, & Ruiz-Cortés, 2017). Since activities in an event log are categorical variables, the task of predicting the next event in a case is solved using classification techniques. A quantitative cross-benchmark of classification techniques for next event prediction is deemed necessary to support researchers and practitioners in choosing the best performing method in a given context. Due to the fact that most researchers in the field of business process management are only relatively familiar with particular machine learning methods, in fact, selecting the best available classification method is often a non-trivial process. Moreover, when the researchers propose a new classifier, they usually exclusively compare its performance with the one of other classifiers within the same family. Therefore, a rigorous quantitative comparison of the performance of classifiers from different families and for different event logs is currently lacking.

It is obvious that some classifiers might have superior performance on particular event log data sets, yet their average performance might be less superior when a higher number of data sets are considered (Macia & Bernadó-Mansilla, 2014). On the contrary, classifiers having low performance on particular data sets may perform significantly better on other data sets with different characteristics. Therefore, the only possible way to assess the actual performance of classifiers with a reasonable amount of confidence is to conduct an experimental benchmark, involving a number of classifiers spanning from different families over a number of data sets. Despite the fact that some classifiers might be 'superior' for particular data sets, the 'superior' performer will vary across data sets, which corresponds to the 'no-free-lunch' theorem (Wolpert, 1996).

The goal of the quantitative cross-benchmark presented in this paper is twofold. First, we aim at identifying the classification algorithm that performs overall best across all event logs considered. This can be helpful to identify a classifier that is likely to perform well in all possible situations, such as when there is no particular information about the nature of an event log. Second, we discuss how the nature of an event log, and in particular its level of variability at the case level, affects the choice of the best classifier. The insights obtained in this case can be used to identify particular classifiers that are likely to perform better with specific event logs.

While previous research has developed quantitative benchmarks of different machine learning techniques for predicting remaining process time (Verenich, Dumas, La Rosa, Maggi, & Teinemaa, 2018) and process outcomes (Teinemaa, Dumas, La Rosa, & Maggi, 2019), this paper proposes an extensive empirical cross-comparison study of machine learning techniques for a different predictive monitoring use case, that is, the next event prediction. In the proposed benchmark, we consider a wide array of classifiers spanning across different families, i.e. tree, Bayesian, rule-based, neural, and ensemble (meta) classifiers. Also, we consider six real-life event log data sets with different characteristics and widely used in the academic community to facilitate future benchmarks. We also consider different re-sampling procedures, i.e. *k*-fold cross validation, repeated *k*-fold cross validation, and repeated hold-out (*subsampling*). Finally, we consider two different sampling strategies, i.e. case-level and event-level sampling.

The remainder of the paper is organized as follows. Section 2 summarizes the related work, while Section 3 describes the settings of the experiments run to build the proposed benchmark. Section 4 discusses the experimentation and the results obtained. Lastly, Section 5 is devoted to conclusions and the discussion of limitations and future work.

2. Related work

A recent survey by Márquez-Chamorro, Resinas, and Ruiz-Cortés (2017) has elaborated a digest of runtime prediction techniques in business processes using event logs. Different kinds of computational predictive techniques, i.e., statistical or based on machine learning, are briefly outlined in the paper. Since in our paper we focus on machine learning techniques for predicting the next event, in this discussion of related work we exclude prediction techniques not implemented using machine learning. We chronologically summarize existing studies about classification algorithms for predictive monitoring in Table 1.

Several machine learning algorithms have been applied for business process predictive monitoring. As far as application scenarios are concerned, process predictive monitoring considers mainly two use cases: (i) predicting process outcomes, such as prediction of service level objectives (SLOs) values, service level agreement (SLA) violations, or linear temporal logic (LTL) constraint violations, and (ii) proactive process monitoring, such as predicting the next event in a case or its timestamp.

As far as machine learning techniques are concerned, decision trees have been adopted in a variety of different works (Conforti, de Leoni, La Rosa, van der Aalst, & ter Hofstede, 2015; Di Francescomarino, Dumas, Maggi, & Teinemaa, 2016; Leitner, Ferner, Hummer, & Dustdar, 2013; Maggi, Di Francescomarino, Dumas, & Ghidini, 2014; Unuvar, Lakshmanan, & Doganata, 2016). Support vector machines (SVM) are the next method most frequently used (Cabanillas, Di Ciccio, Mendling, & Baumgrass, 2014; Kang, Kim, & Kang, 2012; Verenich, Dumas, La Rosa, Maggi, & Di Francescomarino, 2016). Other kinds of predictive techniques have also emerged, such as artificial neural networks (Leitner et al., 2013), deep neural networks (Evermann, Rehse, & Fettke, 2017; Tax, Verenich, La Rosa, & Dumas, 2017), random forests (Leontjeva, Conforti, Di Francescomarino, Dumas, & Maggi, 2015; Teinemaa et al., 2016), and evolutionary computing (Márquez-Chamorro, Resinas, Ruiz-Cortés, & Toro, 2017). Overall, the capabilities of these and other machine learning techniques in process predictive monitoring are still largely undiscovered. In practical settings, it is likely that researchers might not be able to use such classifiers effectively due to lack of knowledge/expertise (parameter tuning, kernel selection, for instance), thereby allowing them to be picked arbitrarily (Fernández-Delgado, Cernadas, Barro, & Amorim, 2014). Unuvar et al. (2016) introduce a procedure for describing five different decision tree models of path followed by a process instance. These models are trained using different tree complexity at each representation and considering the accuracy metric. An object prediction, namely SLOs, is analyzed by Leitner et al. (2013). A decision tree along with neural network and ARIMA model are used to predict nominal SLOs, measurement SLOs, and aggregated measurement SLOs. A decision tree is used to predict the violation of LTL constraints by Maggi et al. (2014). The probability of fulfilling the constraints is calculated roughly by the system by checking compliance with a set of automata representing constraints. A decision support system for predicting the probability of risk in a process case is proposed by Conforti et al. (2015). The proposed system is based on a decision tree algorithm in which several input attributes, i.e., resources, activity duration, frequencies, current workflow, and the next activity are involved. Lastly, a decision tree classifier combined with unsupervised learning algorithm, i.e., clustering, is employed to predict whether the violation of LTL constraints in the work of Di Francescomarino et al. (2016). Process instances are grouped into a cluster based on their similarity in respect of historical traces. A decision tree is then applied to the generated clusters.

Support vector machines (SVM) are considered by Kang et al. (2012) for prediction of process performance and

Table 1
A year-by-year recapitulation of classification techniques applied to predictive monitoring.

Study	Method	Metrics	Prediction label
Kang et al. (2012)	Support vector machine	Error rate	Indicator
Leitner et al. (2013)	Decision tree and neural network	Precision	SLO
Cabanillas et al. (2014)	Support vector machine	*F*-score	Risk
Maggi et al. (2014)	Decision tree	Precision	LTL rules
Breuker, Delfmann, Matzner, and Becker (2014)	Expectation–maximization	Accuracy	Next event
Conforti et al. (2015)	Decision tree	N/A	Risk
Leontjeva et al. (2015)	Hidden Markov Model and Random forest	AUC	Indicator
Unuvar et al. (2016)	Decision tree	Accuracy	Next event
Di Francescomarino et al. (2016)	Decision tree	Accuracy	LTL rules
Teinemaa et al. (2016)	Random forest	*F*-score	Risk
Verenich et al. (2016)	Support vector machine	AUC	Next event
Márquez-Chamorro, Resinas, Ruiz-Cortés, and Toro (2017)	Evolutionary computing	*F*-score	Indicator

Table 2
Process log example for predicting the next event.

case_id	etype_id	complete_timestamp
173688	3	10/01/2011 19:45
173688	5	10/01/2011 20:17
173688	6	10/13/2011 18:37
173691	3	10/01/2011 19:43
173691	5	10/01/2011 22:36
173691	6	10/10/2011 19:30
173691	6	10/10/2011 22:17
…	…	…

of the next event in a case. Attributes such event labels and sequence flow are included to create the prediction model. Cabanillas et al. (2014) recommend a SVM algorithm to classify the successful completion of a process in an air traffic scenario. The aim is to predict the possibility of a flight to show a diversion in the landing. As input attributes, the authors consider domain specific variables such as coordinates, covered distance, and speed of the plane. Verenich et al. (2016) propose to use a SVM classifier to minimize over-processing waste in a business process, by predicting unnecessary knockout checks in process.

A neural-based classification algorithm has been employed by Evermann et al. (2017). A deep learning model based on recurrent neural networks is used to predict the next event in a business process. The study is originally driven by a natural language processing (NLP) problem, whereby the prediction of the next word in a sentence is taken into account. By considering process event logs as text, process traces are then treated as sentences, and process events as words. The work presented by Tax et al. (2017) uses Long Short-Term Memory (LSTM) neural networks to build a predictive model for business process monitoring. The model shows a consistently accurate performance in predicting the next event and the remaining time of a business process.

As a representation of meta-classifier for predictive monitoring, random forests have been used to predict the binary value of a temporal constraint, whether it is fulfilled or not for a determined process instance (Leontjeva et al., 2015). The input features are obtained from different encodings, representing some sort of relevant information, such as order of the events, frequencies of occurrence, and attribute values of the last event. A combination of text mining and sequence classification techniques is proposed by Teinemaa et al. (2016) to improve predictions as structured and unstructured information are merged for the encodings. For the text extraction, the authors apply linear discriminant analysis (LDA) and naive Bayes (NB) techniques, while random forest and logistic regression are used to build the predictive model. Finally, a recent work presented by Márquez-Chamorro, Resinas, Ruiz-Cortés, and Toro (2017) proposes an evolutionary algorithm that outperforms other machine learning techniques, i.e. decision tree, random tree, neural network, and SVM, when it is used for run-time prediction of business process performance indicators.

Given the high number of approaches for process predictive monitoring available, the literature has recently started focusing on frameworks and benchmarks to suggests the best techniques in a given situation. Di Francescomarino, Ghidini, Maggi, and Milani (2018) and Santoso (2018) propose qualitative frameworks for choosing the best technique given the target of predictions and the context of an organization. As mentioned in the introduction, Verenich et al. (2018) and Teinemaa et al. (2019) have developed more quantitative benchmarks of different techniques for predicting remaining time and outcomes of a business process, respectively.

3. Benchmark experiment settings

This section presents the research framework. It comprises detail about data sets preparation, the classification techniques (or classifiers) considered by our comparison benchmark, re-sampling strategies, and statistical significance tests.

3.1. Data sets preparation

The benchmark considers 6 event log data sets. These event logs have been chosen due to their availability and accessibility for further analysis[1]. In addition, these event logs have been considered consistently in previous works. For each event log, we have identified a specific attribute as the *event type* that the prediction task aims at predicting. Note that, in most cases, this event type corresponds to an activity label, i.e., an activity that was executed in a particular instance of a business process. However, in some cases an event type may represent different information that can still be used for process-aware analysis of event logs, such as status of customer applications or claims. In the event logs used in this benchmark, when not explicitly specified, an attribute *activity* was available to be considered as event type. The 6 event logs are presented more in detail in the following:

1. Helpdesk[2]
   The data set includes events from a ticketing management process of the help desk of an Italian software company. The process has 9 distinct event types (activities), and all cases are initiated with the inclusion of a new ticket into the ticketing management system. Each case terminates when the issue is resolved and the ticket is closed. This log contains 3804 process instances and 13,710 events (Tax et al., 2017).

[1] Some data sets could be accessed via *R* interface, see Janssenswillen and Depaire (2017).
[2] https://doi.org/10.17632/39bp3vv62t.1.

2. BPIC 2012[3]
   The data set is publicly available from the Business Process Intelligence Challenge (*BPIC*). It is a processed event log of a loan application in a Dutch financial institution that comprises 262,200 events in 13,087 cases. This event log is available in the standard XES format (IEEE, 2016) and we consider as event type the combination of the classifiers `concept:name` and `lifecycle:transition`, which represents the status of applications. This log contains 36 different event types.
3. BPIC 2013[4]
   The data set represents an incident management log of Volvo IT Belgium. The log contains events from an incident and problem management system called VINST. It contains 7554 cases and 65,533 events. Similarly to BPIC 2012, this log is available in the standard XES format (IEEE, 2016) and we consider as event type is the composition of classifiers `concept:name` and `lifecycle:transition`, which leads to 13 distinct event types.
4. Sepsis[5]
   This real-world event log includes events of sepsis cases from a hospital. One case represents the pathway of one patient through the hospital. The events are collected by the ERP system of the hospital. There are 1050 cases with in total 15,214 events, which are recorded for 16 distinct event types (activities) (Mannhardt & Blinde, 2017).
5. Road Traffic Fine Management[5]
   This data set is a real-life event log of an information system managing road traffic fines. It comprises 34,724 events in 10,000 cases with 11 distinct event types (activities) (Mannhardt, De Leoni, Reijers, & Van Der Aalst, 2016).
6. Hospital Billing[5]
   The data set is gathered from the financial modules of the ERP system of a regional hospital. The event log contains events that are related to the billing of medical services that have been offered by the hospital. Each trace of the event log records the activities executed to bill a package of medical services that are bundled together. It contains 16 distinct event types (activities), 49,951 events, and 10,000 cases (Mannhardt, de Leoni, Reijers, & van der Aalst, 2017).

Formally, an event log L is constituted by a set of traces T, where each trace comprises a list of events. Let A be the universe of log event type labels and let E be the universe of events. An event $e \in E$ comprises several attributes. An event has 3 mandatory attributes, namely a case identifier $\#_{case}(e)$, which maps an event to a case, a timestamp $\#_{tst}(e)$, which logs the time instant at which an event occurred, and an event type label $\#_{ety}(e)$, which, as discussed above, maps an event to the activity or status change that it recorded, i.e., $\#_{ety}(e) \in A, \forall e \in E$. Other attributes describing events may be available. An event log L can be seen simply as a set of events. More specifically, by considering only the event type label attribute for each event, an event log L can be seen as a multiset of traces, where each trace is a sequence of events $e \in E$. A trace t_i is defined as a sequence of events $t_i = [e_{i_1}, e_{i_2}, \ldots, e_{i_n}]$ that were executed in a case, where e_{i_1} and e_{i_n} denote the first and the final event, respectively. That is, all events in a trace have the same case identifier. For example, Table 2 shows sample events for 2 traces with case id 173688 and 173691, respectively.

The preprocessing of the raw event logs for the next event prediction use case is defined in the following. The original event log data sets normally contains 3 features, i.e. case_id, etype_id, and complete_timestamp (see Table 2). Subsequently, it is necessary to preprocess an event log to transform it into a final format that complies with the machine learning algorithm considered. The input attributes of the final data set are composed by the properties of different events and the sum of events duration. One class attribute is also generated. This takes the value of the next event event with respect to the window-size (or prefix length) that has been considered considered. In our case, a 3-sized window is selected, thus the final data set consists of 4 nominal attributes, i.e., case_id, etype_1, etype_2, and etype_3; a numeric attribute, i.e., duration between the first and last event in the considered time window; and a class label attribute, i.e. the next event (next_etype). An example final tabular format of event log used in our experiment is shown in Table 3. To limit the number of possible combinations in our experiments, we have chosen a fixed 3-sized window for all event logs. Previous works, such as Márquez-Chamorro, Resinas, Ruiz-Cortés, and Toro (2017) and Tax et al. (2017), have experimented with window sizes comprised between 2 and 6. The size 3 has been chosen because it is has also been chosen by Márquez-Chamorro, Resinas, Ruiz-Cortés, and Toro (2017) for the BPIC 2013 data set and other event logs after an analysis of performance of different window sizes in the case of outcome-based predictive monitoring (Márquez-Chamorro, Resinas, Ruiz-Cortés, & Toro, 2017).

Table 3
Final format of event log example shown in Table 2.

case_id	etype_1	etype_2	etype_3	Duration	next_etype
173688	3	5	6	17,212	3
173691	3	5	6	12,947	6
173691	5	6	6	12,941	6
…	…	…	…	…	…

3.2. Classification methods

We employ 20 classifiers implemented in *R* and *Weka* (Hall et al., 2009). All the implementations used in this paper are free software. A software package in *R*, called *mlr* (Bischl et al., 2016), is used as interface for the experiment as it enables to set *hyperparameter* values and selecting different re-sampling methods (*k*-fold cross validation, repeated cross validation, and repeated hold-out). Some classifiers implemented in *Weka* are used by running the command-line version of the *java* class for each classifier. We briefly describe the twenty classifiers in the remainder of this section.

1. Decision tree (*DT*)
   A top-down approach is commonly used for tree construction. The tree is made up of root and some nodes, while the choice of the feature for a node is based upon the impurity of the distribution of the class label. The impurity could be measured in two different ways, i.e., Gini index or entropy-based. Tree-pruning strategy is suggested to obtain a generalized sub-tree during the growing phase and to avoid from *overfitting* problems. In our experiment, we use the *C*4.5 algorithm, which is one of the alternatives for constructing the tree. It is implemented in Weka as *J*48 classifier (Quinlan, 1993).
2. Credal decision tree (C-DT)
   Unlike *J*48, which exploits information gain as split criterion to select the split attribute at each branching node, C-DT considers imprecise probabilities and the application of uncertainty measures for the original split criterion (Abellán & Moral, 2003). It uses reduced-error pruning (with back-fitting) and sorted values for numeric attributes. Missing values and numeric at-

[3] https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.
[4] https://doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee.
[5] https://www.bupar.net/eventdataR.html.
[5]
[5]

tributes are handled like *C*4.5. Learning parameter settings of C-DT includes fold numbers, which specifies the amount of data used for pruning, the root attribute, which is used to fix the root node of the tree, and the *S* value, which is a parameter in the Imprecise Dirichlet Model to obtain imprecise probabilities as intervals. A *java* implementation of C-DT in Weka is used in our experiment.

3. Random tree
   A random tree (*RT*) differs strikingly from the standard tree training in feature splitting. The tree is grown using *K* randomly chosen features at each node with no pruning (Breiman, 2001). A *RT* implementation in Weka allows to estimate class probabilities based on an hold-out set or back-fitting.
4. Decision stump
   A decision stump (*DS*) is a decision tree with one root that is directly linked to its leaves. Due to its simplicity, the DS classifier can also be considered as a 1-level decision tree (Iba & Langley, 1992). It is commonly used as a base classifier in ensemble learners, i.e. bagging and boosting. However, we herein are only interested to assess its performance as an individual classifier. We use the implementation of decision stump in Weka.
5. OneR (1-R)
   This classifier is similar to decision stump, which is a straightforward classifier that produces one rule for each predictor in the data set, and then choose the rule with minimum total error as its final rule. A rule for a predictor is generated by constructing a frequency table for each predictor variable against the target variable (Holte, 1993).
6. Conjunctive rule (*CR*)
   A conjunctive rule classifier produces rules, consisting of antecedents and consequents. The consequent is the distribution of the number of classes in the data set. The classifier chooses an antecedent by calculating information gain of each antecedent and prunes the resulted rule using reduced error pruning. This strategy reduces the complexity of the final classifier since the rule that gives little prediction is removed, which, consequently enhances accuracy (Friedman, Hastie, & Tibshirani, 2001).
7. Ripper
   This classifier has been initially proposed with the aim of improving the *IREP* algorithm (Cohen, 1995). The classifier builds a rule by using the following two strategies: (1) instances are randomly divided into two subsets, i.e. a growing set and a pruning set, and (2) a rule is grown using the FOIL algorithm. After constructing a rule, the rule is directly pruned by removing any final sequence of conditions from the rule. We use the *java* class implementation of this classifier in Weka as *JRip*.
8. Naive Bayes (*NB*)
   A Naive Bayes (*NB*) classifier considers the conditional probabilities of a categorical class feature specified using independent predictor features using the Bayes rule (John & Langley, 1995). A standard *NB* classifier that uses estimator precision values picked from the analysis of training data is considered. It assumes independence of the predictor variables, and, for features with missing values, the corresponding instances are ignored for prediction.
9. Support Vector Machine (*SVM*)
   A support vector machine builds a set of *hyperplanes* in a higher dimensional space used for classification and regression. It was firstly introduced by Cortes and Vapnik (1995). We consider a *L*2-regularized *L*2-loss support vector classification with descent methods implemented in *LIBLINEAR* (Fan, Chang, Hsieh, Wang, & Lin, 2008). Since we deal with a large number of features, a linear classification is tailored with such data. Linear *SVM* is utilized as it supports multi-class classification problem. Also, the considered implementation is highly efficient for training large-scale problems as compared, for instance, to *LIBSVM* (Chang & Lin, 2011). The training parameters are set as follows: cost $C = 100$, $\epsilon = 0.001$, and maximum iteration is 1000.
10. Deep neural network (*DNN*)
    We use R interface for H_2O, an open source deep learning framework for big data analysis (Candel, Parmar, LeDell, & Arora, 2016). The H_2O deep learning is derived from a multi-layer feed forward neural network that is developed using stochastic gradient descent of back-propagation. The main difference from standard neural networks is that it considers a large number of hidden layers. In order to enable high predictive accuracy, some parameter settings, such as activation function, adaptive learning rate, rate annealing, l_1 and l_2 regularization, and number of hidden layers are obtained using a grid search. The number of hidden layer is 3, with 100 nodes at each layer.
11. Long Short-Term Memory (*LSTM*)
    LSTM (Hochreiter & Schmidhuber, 1997) is commonly utilized to modeling sequential data. The building block of LSTM is a memory cell, which fundamentally depicts the hidden layer. As the next event prediction can be considered as a sequence prediction, LSTM can be adopted in this task. To conform with the input of other classifiers in the experiment, we run a sequence modeling task, so-called *many-to-one* (Karpathy, 2015), where the input data is a sequence of 3 event types (ie., considering a window size of 3) and the output is the next event type. The duration attribute is not considered for this classifier since it applies to the entire window and not to each individual event, i.e., it is not sequential. We use the standard implementation of LSTM in *TensorFlow*.
12. Bagging
    The bagging ensemble takes bootstrap replicates of the learning set as new learning sets. A data set is partitioned into several subsets and the base classifier is trained on each subset. The final prediction is obtained by average aggregation when predicting a numerical output and a considering a majority vote when predicting a class (Breiman, 1996). It can be used for classification and regression depending on the base learner. We use a weak classifier, *J*48, as the base model of bagging.
13. Boosting
    Unlike bagging, the boosting ensemble treats base classifier in a sequential way. A set of classifiers is trained sequentially and the results of a previous classifier are incorporated into the next one to improve the final prediction by having the latter one emphasizing more on the mistakes of the earlier classifiers (Freund & Schapire, 1997). In our experiment, we employ the Adaboost algorithm (Freund and Schapire, 1996) and *J*48 as a base classifier. An implementation in Weka workbench is chosen to run such ensemble scheme.
14. Multiboost
    This meta-classifier is an expansion of the highly successful Adaboost method for organizing ensemble committees. It is formed of Adaboost and wagging, thus enabling to exploit both high bias and variance reduction. Using *J*48 as the base learner, Multiboost has outperformed Adaboost wagging significantly in terms of lower error rate over a large representative data sets (Webb, 2000).
15. Dagging
    The Dagging ensemble learner produces a number of disjoint, stratified folds out of the data set and sends each partition of data to the base classifier. Final predictions are established via majority voting (Ting & Witten, 1997). We use *J*48 as base classifier and the number of folds is 10.
16. Random subspace (*RS*)
    Firstly introduced by (Ho, 1998), the random subspace classifier is also called feature bagging, as it is very similar to bagging,

except that the feature are randomly sampled. It tries to reduce the correlation between estimators by training them on random feature samples with replacement instead of the full feature set. It combines the final outputs of the individual models by majority voting. A decision tree classifier (*J*48) is commonly used as a base model.

17. Nested dichotomies (*ND*)
The nested dichotomies is a meta-classifier for handling multi-class data sets by constructing a random tree structure (Frank & Kramer, 2004). It provides a statistically way of applying binary class learning algorithms to multi class problems. In order to produce more accurate classifications, it is recommended to use *J*48 or logistic regression as base classifier. The nested dichotomies with *J*48 has demonstrated a better performance than applying *J*48 directly to multi-class problems (Dong, Frank, & Kramer, 2005).
18. Decorate
Decorate is a meta-learner for constructing various ensembles of classifiers by using artificial training examples (Melville & Mooney, 2003). At each step, some artificial samples are randomly generated and fed into the original training set in order to build a new ensemble member. Extensive experiments have revealed that this technique has performed significantly better than base classifiers, Bagging ensembles, and random forests (Melville & Mooney, 2005). We consider *J*48 in Weka as a base model of this ensemble scheme.
19. Random forest (*RF*)
Random forest classifier is built based on a forest of classification and regression trees, allowing to reduce variance. Each tree is a weak classifier built on a subset of rows and columns. An average prediction over all the trees is considered as the final prediction (Breiman, 2001). In our experiment, we employ a distributed random forest framework implemented in H_2O through the *R* interface. A grid search is applied in order to find the optimum value of training parameters, i.e. maximum depth, sample rate, number of bins, and histogram type. We set a large number of trees (1000) to build the forest.
20. Gradient boosting machine (*GBM*)
Similar to random forest, gradient boosting machine (*GBM*) takes classification and regression tasks at once. It is one of the homogeneous ensemble learners, where the same type of weak classifier models are considered to form a final prediction. The trees are grown sequentially, with later trees depending on the results of previous trees (Friedman, 2001). In order to get a reasonable classification accuracy, *GBM* requires tuned *hyperparameters* by applying a *grid* search. We employ the *GBM* implementation in H_2O framework via *R* interface. Training parameters of GBM include maximum depth, minimum rows, number of bins, learning rate, learning rate annealing, sample rate, and histogram type. A large number of trees is also set to 1000.

3.3. Validation procedures

Different types of validation techniques have been considered in order to minimize the variability of each data set. Also, validation procedures are used to prove that the classifier's accuracy is not obtained by chance. We briefly explain the validation methods used in our experiment in the following.

- *Cross validation*. It is a validation technique that uses different subsets of samples with no overlapping among themselves. For *k*-fold cross validation, the data set $\mathcal{G}$ is partitioned into *k* disjoint subsets $\mathcal{F}_j$ of equal size, such that $\bigcup_j \mathcal{F}_j = \mathcal{G}$. In the *n*th of the *k* iterations, the *n*-subset is used as testing set, while the merge of the remaining subsets is considered as training set. We consider a 10-fold cross validation (*kf*10).
- *Repeated cross validation*. It is performed by considering five repetitions of a 2-fold cross validation (*kf*2), that is, $(5 \times kf2)$, where we are able to obtain five training and five testing subsets at 50%. In this case, samples can overlap, but it is less strikingly than in the *kf*10.
- *Hold-out*. It performs a validation procedure in such a way that the data set $\mathcal{G}$ is randomly partitioned into a training and a testing set according to a given ratio, i.e., 80% training and 20% testing set. In order to obtain the same number of elements as in *kf*10 and $5 \times kf2$, we replicate this procedure 10 times $(10 \times ho)$ and consider the average performance.

As far as sampling is concerned, observations in the pre-processed datasets are represented by prefixes of length 3 (see Table 3) generated from traces in the event log. We consider two different sampling approaches of prefixes, namely the *event-based* and *case-based* sampling. In the event-based sampling, prefixes in the pre-processed event log are sampled without considering the value of the *case_id* attribute, that is, the training and testing sets may contain prefixes generated from the same trace. In the case-based sampling, information about *case_id* is maintained while sampling prefixes, which means that all prefixes generated from a given trace belong to either the training or the testing set. Our objective is to assess whether the sampling strategy bears a significant impact on the performance of classifiers.

3.4. Significance tests

Following a recommendation of Demšar (2006), we use several significance tests to compare whether there exist differences in performance accuracy of the considered classifiers. We use the following tests to benchmark multiple classifiers over multiple data sets:

- *Friedman test* (Friedman, 1940). It analyzes the classification algorithms based on the ranks of each algorithm on each data set. Let us suppose to have *n* data sets and *m* classifiers, each algorithm is ranked (in ascending order) for each data set individually with respect to the performance accuracy *pa*. For data set D_i, the classifier c_j such that $pa_{ij} > pa_{ij'}, \forall j', j, j' \in \{1, 2, \dots, k\}, j \neq j'$, is ranked 1. If there is a δ-way tie after the rank *r*, the rank $[(r+1)+(r+2)+\dots+(r+\delta)]/\delta$ to each of the tied classifiers is considered. Let r_i^j be the rank of classifier c_j on data set D_i, then the average rank of classifier c_j on all data sets is $\bar{R}_j = \frac{1}{n}\sum_{i=1}^{n} r_i^j$. Under the null hypothesis, i.e., all classifiers perform equivalent, the Friedman statistic is calculated as:
$$\chi_F^2 = \frac{12 \times n}{m(m+1)}\left\langle \sum_{j=1}^{m} R_j^2 - \frac{m(m+1)^2}{4} \right\rangle \tag{1}$$
which is distributed as χ_F^2 with $m-1$ degrees of freedom.
- *Iman–Davenport test*. Iman and Davenport (1980) found that Friedman χ_F^2 present a conservative behavior and a better statistic could be calculated as:
$$F_F = \frac{(n-1)\chi_F^2}{n(m-1)\chi_F^2} \tag{2}$$
which is distributed as the *F* distribution with $(m-1)(n-1)$ degrees of freedom.
- *Nemenyi test*. Nemenyi (1962) proposes a test that is carried out after Friedman test if the latter rejects the null hypothesis. The Nemenyi test compares two classification algorithms based on the assumption that the performance of two classifiers are significantly different if the corresponding average ranks differ

Table 4
Average accuracy (in %) and Friedman ranking for each classifier with the three different validation methods in terms of case-based sampling.

Classifier	Validation methods						Average	
	*kf*10		5 × *kf*2		10 × *ho*			
	Acc.	Rank	Acc.	Rank	Acc.	Rank	Acc.	Rank
Decision tree	72.71	5.42	72.02	5.33	72.79	5.17	72.50	5.31
Credal decision tree	**73.75**	**3.42**	**73.51**	**2.17**	**73.62**	**2.33**	**73.63**	**2.64**
Random tree	57.61	16.33	56.07	16.33	59.17	15.83	57.62	16.17
Decision stump	46.99	18.17	46.98	18.25	47.02	18.42	47.00	18.28
OneR	65.68	12.50	65.71	12.33	65.68	12.33	65.69	12.39
Conjunctive rule	46.99	18.33	46.86	18.42	46.97	18.58	46.94	18.44
Ripper	70.41	8.75	70.31	8.83	70.47	8.67	70.40	8.75
Naive bayes	64.23	14.50	63.81	14.67	63.82	15.50	63.95	14.89
Support vector machine	46.92	18.00	44.97	18.17	44.60	18.33	45.50	18.17
Deep neural network	71.65	8.50	71.08	10.00	72.36	7.33	71.69	8.61
Long Short-Term Memory	67.72	11.83	64.06	13.17	68.22	12.00	66.67	12.33
Bagging	70.77	8.33	71.03	6.50	71.10	7.83	70.97	7.56
Boosting	57.14	11.75	58.05	12.33	59.25	12.17	58.15	12.08
Multiboost	70.08	9.92	69.19	10.50	69.94	10.08	69.74	10.17
Dagging	69.45	7.67	71.96	6.33	71.96	6.92	71.12	6.97
Random subspace	72.03	8.83	72.18	7.33	72.32	7.67	72.18	7.94
Nested dichotomies	71.81	6.83	71.13	7.83	72.01	7.50	71.65	7.39
Decorate	70.78	6.83	70.55	7.67	70.61	7.50	70.65	7.33
Random forest	73.34	3.33	73.23	3.17	72.97	4.83	73.18	3.78
Gradient boosting machine	69.83	10.75	69.25	10.67	70.00	11.00	69.69	10.81

The best value is indicated in bold.

more than or equal to a critical difference (*CD*), which is computed as:

$$CD = \frac{q_\alpha}{\sqrt{\frac{m(m+1)}{6\times n}}} \tag{3}$$

The q_α denotes the q value of Tukey test, see Table A.8 in Japkowicz and Shah (2011), weighed by dividing it by $\sqrt{2}$.

We are also interested in calculating the p-value adjustment, which denotes the lowest level of significance. For this purpose, we can discover whether two algorithms are significantly different in terms of a metric and how large this difference is. We consider two procedures to calculate "adjusted p-values" (Wright, 1992). These are chosen due to their simplicity and high power when comparing all classifiers with one chosen as control:

- *The Holland test* (Holland & Copenhaver, 1987). It is a step-down adjusted α value procedure. Let p-values be ordered as $p_1, p_2, \ldots, p_{m-1}$ so that $p_1 \le p_2 \cdots \le p_{m-1}$ and $H_1, H_2, \ldots, H_{m-1}$ denote the corresponding hypotheses. Holland rejects H_1 to H_i if i is the smallest integer such that $p_i > 1-(1-\alpha)^{m-i}$. If p_1 is below $1-(1-\alpha)^{m-i}$, then the corresponding hypothesis is rejected and the test continues with the second by comparing p_2 with $1-(1-\alpha)^{m-i}$, and so forth.
- *The Rom test* (Rom, 1990). It is a step-up test procedure working in the opposite direction of the Holland test, comparing classifiers with the largest p value with α, the second largest with $\alpha/2$, the third with $\alpha/3$, and so on until it encounters a hypothesis that it can reject.

4. Experiment results and discussion

In the proposed experimental benchmark, we assess 20 classifiers over 6 event log data sets, 2 sampling strategies, and 3 different validation techniques, which lead to a total of 720 combinations. The pre-processed event logs, as well as experiment results are publicly available for future benchmark and research.[6] The test results are the average of 10 replications at each validation method (as we specify in Section 3.3). We use those validation methods in order to minimize poor bias and variance due to small data sets included in this experiment. In addition, different validation methods ensure that the classifier's accuracy is not obtained incidentally.

We report the results of Friedman ranking in order to statistically calculate the classifiers rank with respect to the all event log data sets. The ranking is an extremely valuable tool in benchmarking classifiers as it reflects an order of their performances (the ranking is getting better as the classifier's performance accuracy increases). Tables 4 and 5 show the average accuracy and Friedman rank over the six event logs for case and event-based sampling, respectively.

The remainder of this section discusses first the overall results obtained. In particular, the objective is to understand which is the best classifier for the next event prediction task across all data sets and whether the ranking of classifiers obtained is significant from a statistical point of view. Then, we analyze in a more qualitative way how the *context*, that is, the characteristics of an event log, influences the choice of best-performing classifier.

4.1. Overall results analysis

For case-based sampling, the best classifier is credal decision tree (C-DT), with average rank 2.64 and average accuracy 73.63%, followed by random forest (RF), with average rank 3.78 and average accuracy 73.18%. It is interesting to note that C-DT maintains its performance, regardless of the validation method used. It is also worth mentioning that, as a single classifier, C-DT is able to outperform other ensemble classifiers, such as RF and gradient boosting machine (GBM). For event-based sampling, the results are somewhat similar to the case-based sampling strategy. The highest average ranked classifier is C-DT, with average rank 2.31 and average accuracy 73.60%, followed by RF with average rank 3.75 and average accuracy 73.30%. The C-DT has consistently performed best than other competitor classifiers in terms of all validation techniques.

From a general point of view, an impressive result has been delivered by C-DT, even though it is a single classifier. The RF classifier, which has been widely known for many years as an ensemble

[6] https://github.com/bayuat/Predicting-the-next-event-in-business-process and https://sites.google.com/view/bayuat/research-files.

Table 5
Average accuracy (in %) and Friedman ranking for each classifier with the three different validation methods in terms of event-based sampling.

Classifier	Validation methods						Average	
	*kf*10		5 × *kf*2		10 × *ho*			
	Acc.	Rank	Acc.	Rank	Acc.	Rank	Acc.	Rank
Decision tree	72.53	6.25	72.26	5.67	72.53	4.25	72.44	5.39
Credal decision tree	**73.77**	**2.33**	**73.56**	**2.50**	**73.48**	**2.08**	**73.60**	**2.31**
Random tree	61.51	16.00	52.03	17.33	56.28	17.17	56.61	16.83
Decision stump	47.01	18.67	46.97	18.08	46.95	18.00	46.98	18.25
OneR	65.75	12.25	65.73	12.50	65.60	12.25	65.69	12.33
Conjunctive rule	47.01	18.67	46.92	18.25	46.95	18.17	46.96	18.36
Ripper	70.05	9.50	70.18	9.58	70.30	9.00	70.18	9.36
Naive bayes	64.66	14.50	64.76	14.00	64.71	14.50	64.71	14.33
Support vector machine	48.77	17.83	44.50	18.17	45.88	18.00	46.38	18.00
Deep neural network	72.00	8.33	72.29	7.67	70.41	10.17	71.56	8.72
Long Short-Term Memory	66.11	13.00	64.19	13.50	67.94	12.33	66.08	12.94
Bagging	64.78	9.92	70.84	7.83	70.65	8.58	68.75	8.78
Boosting	59.86	11.08	58.65	12.67	55.65	13.42	58.05	12.39
Multiboost	70.81	9.67	68.87	11.17	70.13	8.92	69.94	9.92
Dagging	72.25	5.50	71.87	6.67	71.79	6.67	71.97	6.28
Random subspace	71.79	8.67	72.17	7.50	72.21	7.67	72.06	7.94
Nested dichotomies	72.10	7.17	72.12	6.75	71.33	7.17	71.85	7.03
Decorate	69.89	7.17	69.95	7.83	69.50	7.25	69.78	7.42
Random forest	73.43	4.00	73.43	3.17	73.03	4.08	73.30	3.75
Gradient boosting machine	70.06	9.50	70.44	9.17	66.89	10.33	69.13	9.67

The best value is indicated in bold.

learner, has performed better than other more recently developed classifiers, such as GBM and deep neural networks (DNN). This is because the training parameters of GBM and DNN may not have been optimally obtained using the *grid* search. In addition, dagging ensemble is among the best classifier ensemble, with average rank 6.97 and 6.28 for case and event-based sampling, respectively. However, its performance is not better than decision tree (DT). This result is surprising given the fact that DT is a base classifier of dagging ensemble. Therefore, the implementation of dagging ensemble has not brought a substantial improvement over the base classifier DT. It is also interesting to note that one of the worst performer is support vector machine. This result points to the importance of optimizing the SVM *hyperparameters*, such as the regularization parameter and the kernel function.

It is also interesting to discuss the best classifier in each family of classifiers, i.e. trees, Bayesian, rule-based, neural and meta classifiers. To do this, the best-in-family classifier is chosen for each family of classifiers, leading to six classifiers for further comparison analysis. Figs. 1 and 2 illustrate the performance comparison of the selected classifiers for different sampling approaches over the six event log data sets and the three validation methods, respectively. From Fig. 1, we can conclude that the performance of SVM and DNN vary with respect to the sampling used, while the performance change of other classifiers, i.e., C-DT, Ripper, RF, and naive Bayes (NB) are very small when varying the sampling and validation methods.

Considering Fig. 2 about the performance accuracy, C-DT obtains good results when it is applied on BPI2012, Sepsis, Road Traffic, and Hospital Billing, while RF obtains good results when it is applied on Helpdesk and BPI2013 Incidents. C-DT mostly performed better with any given event log. We can thus conclude that C-DT is closer to a stable learner for task of next event prediction as this classifier is less sensitive to different perturbation on event log samples. SVM, on the other hand, is a highly unstable learner since its performance relies on the number of event log samples. This leads us not to choose such stable learner for base classifier in ensemble because combining them would not help enhance the generalization performance.

For the sake of a comprehensive comparison, we conduct several statistical tests in order to measure whether the performance differences among the best-in-family classifiers are significant. The results of Friedman and Iman–Davenport tests are reported in Table 6. With 6 classifiers and 6 event log data sets, F_F is distributed with respect to the F distribution with degrees of freedom $df_1 = 6 - 1 = 5$ and $df_2 = (6 - 1) \cdot (6 - 1) = 25$, the p-values for each sampling and validation approaches are determined using the $F(5, 25)$ distribution. We can see that both Friedman and Iman–Davenport reject the null hypothesis (all classifiers perform equivalent) at a high level of significance ($\alpha < 0.01$). Hence, it is necessary to conduct multiple post-hoc tests.

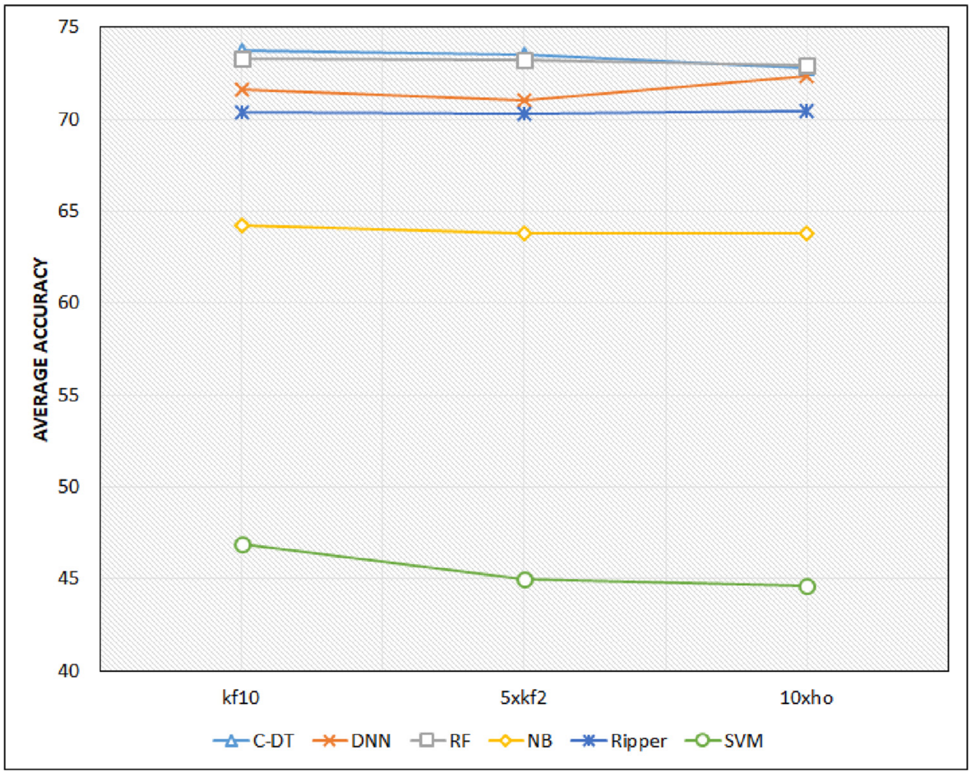
In this study, we pick the best performing classifier (C-DT) as the control classifier for being benchmarked against the other classifiers. Tables 7 and 8 show the adjusted p-values for each validation and sampling approach using Holland (Holland & Copenhaver, 1987) and Rom (Rom, 1990) tests. We set the significance level to $\alpha = 0.1$. Based on these results, we can conclude the following:

- C-DT shows a highly significant difference in respect of SVM, Ripper, and NB, irrespective of the statistical tests;
- C-DT is statistically better than any other classifiers. However, in some cases (i.e. *kf*10 and 10 × *ho*), there are no performance differences between C-DT and the other top-2 classifiers, i.e. DNN and RF, according to the Rom test. Here, Rom test provides a more meaningful result than Holland test.
- In 5 × *kf*2, the performance of C-DT and RF are statistically identical, regardless of the post-hoc tests.

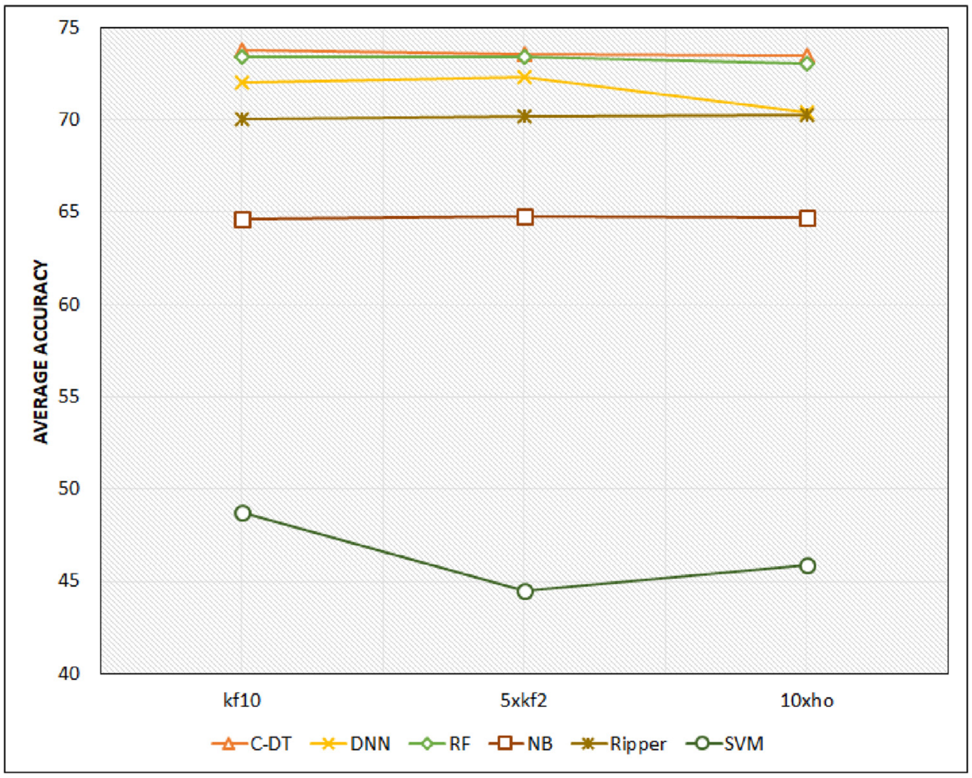
4.2. *Effect of event log type on the choice of classifier*

In order to qualitatively evaluate how the nature of an event log and, consequently, the type of process generating it, affects the performance and choice of classifier, we need to define a criterion to cluster the event logs used in the proposed benchmark into different groups.

We consider variability at the case level to cluster event logs. Specifically, variability at the case level is strictly related with the number of case variants in an event log, i.e., the number of differ-

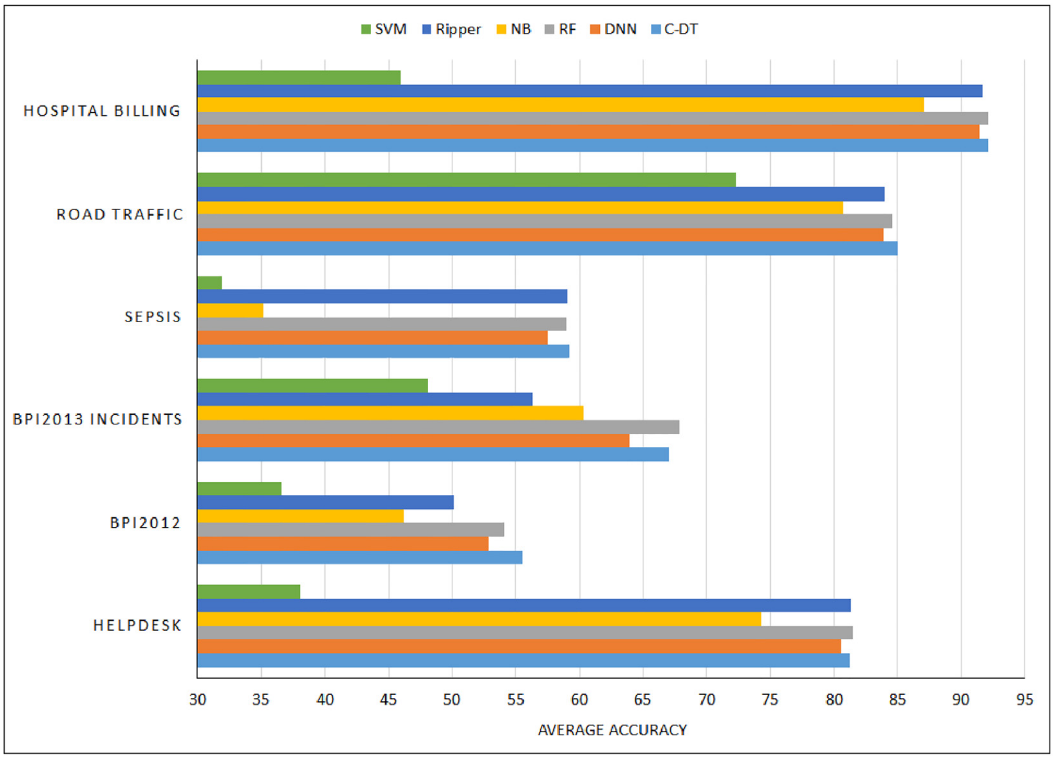


(a) Case-based sampling

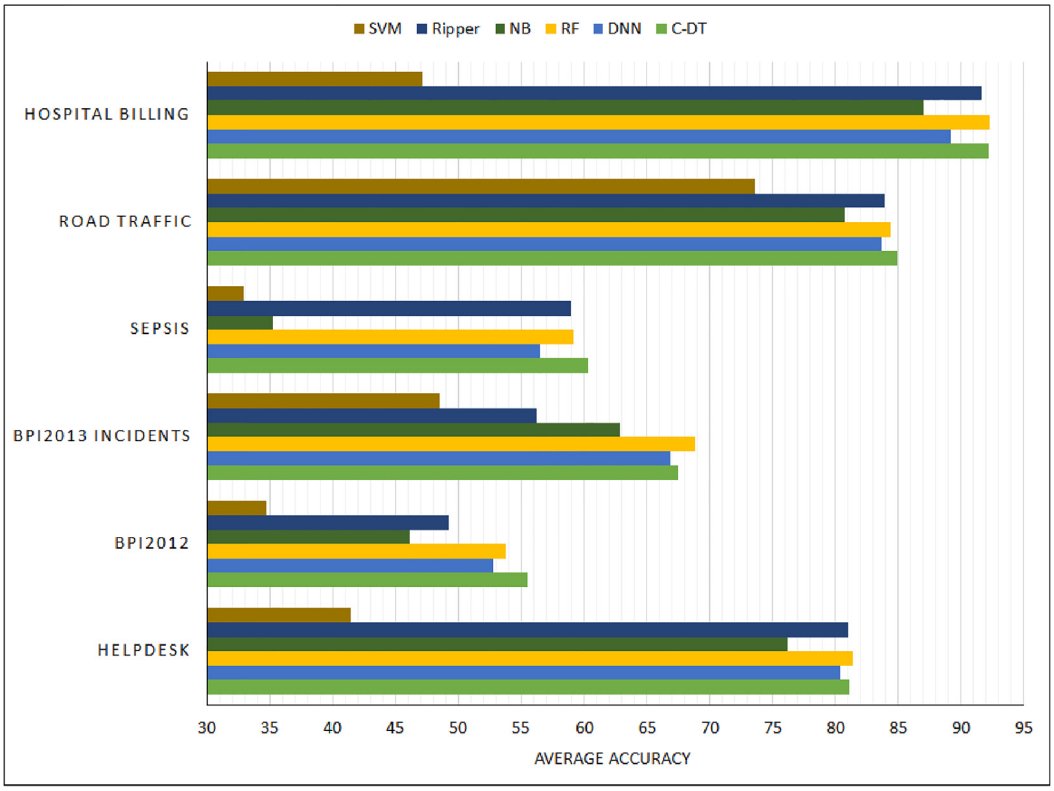


(b) Event-based sampling

Fig. 1. Average accuracy (in %) over the five event log data sets.



(a) Case-based sampling



(b) Event-based sampling

Fig. 2. Average accuracy (in %) over the three validation methods.

Table 6
Results for Friedman and Iman–Davenport tests of the selected classifiers.

Sampling	Validation methods	Friedman test		Iman–Davenport test		H_0 rejection
		χ_F^2	p-value	F_F	p-value	
Case-based	*kf*10	25.71	1.01E−04	30.00	8.46E−10	Yes
	5 × *kf*2	27.52	4.51E−05	55.58	9.78E−13	Yes
	10 × *ho*	21.52	6.45E−04	12.70	3.35E−06	Yes
Event-based	*kf*10	25.91	9.31E−05	31.63	4.84E−10	Yes
	5 × *kf*2	27.62	4.32E−05	58.00	6.02E−13	Yes
	10 × *ho*	26.10	8.55E−05	33.42	2.70E−10	Yes

Table 7
Adjusted p-values for case-based sampling (C-DT is the control classifier).

Classifier	10*kf*			5 × *kf*2			10 × *ho*		
	Unadjusted p	p_{Holl}	p_{Rom}	Unadjusted p	p_{Holl}	p_{Rom}	Unadjusted p	p_{Holl}	p_{Rom}
DNN	0.08963	0.18018	0.18986	0.04486	0.12864	0.13293	0.44040	0.82476	1
RF	0.87737	0.87737	0.87737	1	1	1	0.87737	0.87737	1
NB	0.00548	0.02173	0.02155	0.00203	0.00809	0.00798	0.01355	0.05313	0.05331
Ripper	0.06408	0.18018	0.18986	0.06408	0.12864	0.13293	0.44040	0.82476	1
SVM	0.00006	0.00030	0.00030	0.00003	0.00015	0.00015	0.00039	0.00193	0.00190

Table 8
Adjusted p-values for event-based sampling (C-DT is the control classifier).

Classifier	10*kf*			5 × *kf*2			10 × *ho*		
	Unadjusted p	p_{Holl}	p_{Rom}	Unadjusted p	p_{Holl}	p_{Rom}	Unadjusted p	p_{Holl}	p_{Rom}
DNN	0.08963	0.17123	0.17927	0.08963	0.18018	0.18986	0.02064	0.06064	0.06115
RF	0.87737	0.87737	0.87737	0.75762	0.75762	0.75762	0.64343	0.64343	0.64343
NB	0.00119	0.00477	0.00469	0.00337	0.01341	0.01326	0.00203	0.00809	0.00798
Ripper	0.04486	0.12864	0.13293	0.06408	0.18018	0.18986	0.06408	0.12405	0.12816
SVM	0.00006	0.00030	0.00030	0.00006	0.00030	0.00030	0.00002	0.00008	0.00008

Table 9
Event log classification based on variability level.

Event log	Event types	Trace variants	Most frequent variants to 80% cases	Variability
Helpdesk	9	154	5	Low
Hospital Billing	16	288	3	Low
Road Traffic	11	44	2	Low
Sepsis	16	846	635	High
BPI2013	13	2278	767	High
BPI2012	36	4366	1748	High

ent ways in which an individual case can be executed. However, simply counting the number of trace variants is not likely to be a good indicator of variability, since all real world event logs tend to have a high number of trace variants, due to infrequent behavior and possibly incompletely logged cases. We argue that the number of most frequent trace variants required to cover a sufficiently large portion of the case variability in an event log gives gives a better indication of event log variability.

Table 9 shows, for each event log, the number of event types in the log, the number of trace variants, and the number of most frequent trace variants that cover 80% of the cases in the log. Based on the figures shown in this table, we can clearly partition the 6 logs into 2 clusters, one characterized by *low* variability, comprising logs Helpdesk, Road Traffic and Hospital Billing, and one by *high* variability, comprising BPI2012, BPI2013 and Sepsis logs.

In a nutshell, in low variability event logs, a large majority of cases are executed following a limited number of trace variants, i.e., a limited number of possible sequences of activities. The opposite happens for the high variability event logs, where a very high number of trace variants are required to cover 80% of the process cases. With reference to the process model that can be mined from an event log, by borrowing a commonly accepted jargon in the process mining community, we can refer to *high* variability logs as generating *spaghetti* (i.e., very complex, often unusable) process models, and *low* variability logs as generating *lasagna* (i.e., more regular, simple, understandable) process models (van der Aalst, 2011).

By considering the results on average accuracy of per-family top performing classifiers in Fig. 2, we can draw the following qualitative considerations as far as choosing the best classifier for the next event prediction task is concerned:

- The performance of classifiers is on average much higher in the case of low variability logs as compared to the case of high variability. This is expected, since it should be easier for a classifier to learn the limited number of frequent control flow patterns characterizing the process generating a low variability event log and, therefore, become more accurate in the classification task;
- There is no clearly superior classifier among the top 3 performers for low variability event logs, namely C-DT, Ripper and RF. The performance of DNN in this case is also close to the top 3 performers. This points to the conclusion that the choice of best classifier is less critical for low variability logs, since classifiers from different families perform well and similarly. Still, however, tree-based classifiers (C-DT and RF) show the best performance;

- The superiority of tree-based classifiers is more prominent for high variability event logs, where the performance of DNN and, in particular, Ripper, decreases when compared to low variability logs. This highlights that the overall superiority of tree-based and, specifically, C-DT highlighted in the previous section is mainly due to the fact that they are able to perform better when dealing with high variability logs;
- Other classifiers, such as SVM and NB, perform poorly across both categories of event logs. We cannot conclude whether these types of classifiers are not fit for this classification task, since their performance may be increased by different design choices and *hyperparameter* settings, e.g., different kernel functions in SVM, which have not been tested in this paper.

To sum up, the choice of best classifier appears to be less critical in the case of low variability logs, whereas, for high variability logs, tree-based classifiers C-DT and RF appear to be superior.

5. Conclusions

This paper has presented a quantitative benchmark of different classifiers for classification tasks in next event prediction, a typical use case of business process predictive monitoring. This benchmark complements and improve the previous work of Márquez-Chamorro, Resinas, and Ruiz-Cortés (2017), where a qualitative research survey on business process predictive monitoring has been carried out. A qualitative analysis of the impact of different characteristics of the process generating an event log on the choice of the best classifier for next event prediction has also been discussed.

In the experiment, we have included 20 classification techniques, ranging from diverse families, i.e., trees, bayesian, rule-based, neural and meta classifiers. Furthermore, six real-life event log data sets, two different sampling approaches, as well as three kinds of validation methods were also taken into account. Based on our experiment results, the credal decision tree (C-DT) has emerged as a classification algorithm that is most likely to have an excellent performance for the next event prediction in business process, followed by random forests, decision tree, dagging, and nested dichotomies. Surprisingly, support vector machine have shown a very poor performance.

The work presented in this paper has several limitations. When automated grid search for hyperparameter optimization was not possible, we have relied on standard settings for the considered classifiers. This entails that many of the 20 classifiers have been used considering standard settings and we cannot conclude whether different settings untested in this paper may lead to improved performance of the currently poor performing classifiers. Note, however, that in some cases one may be forced to use standard classifier settings, owing, for instance, to lack of time to explore other settings in a systematic way.

We also use only one trace encoding technique using a fixed sized window. The number of event logs considered in this paper is also limited. A deeper investigation of different classifier design settings, event logs, and trace encoding techniques, however, would have dramatically extended the scope of this paper, making the comparisons, in particular the statistical testing, practically unattainable.

Among the possible ways of extending the scope of this paper, we believe that considering a larger set of event logs with different features is the most interesting one, because it may give more practical insights about choosing the best classifier in different practical settings. Another limitation lies with the lack of focus on the quality of a prediction. In this paper we focus on accuracy of classifiers, but the quality of a prediction, for instance in terms of case-based or time-based stability, should also be an important criterion while choosing a classifier. Decision makers should prefer classifiers that are reliably correct in their prediction over different process cases or in time instead of classifiers that are high performing only in selected process cases or for which predictions vary considerably as new observations become available.

For future work, it would also be worthwhile to address the main limitations emerged in our study about SVM's accuracy, as well as the performance behavior of ensemble learners with different base learner for predicting the next event.

Authors contributions

Marco Comuzzi has supervised the development of the framework, analysis of the results and the writing of the paper.

Bayu Adhi Tama has contributed mainly to the design and implementation of the framework and has taken care of running the experiments.

Acknowledgments

This work has been partially funded by NRF Korea project number 2017076589. The authors thank Jonghyeon Ko and Jongchan Kim for their help with running some experiments.

References

van der Aalst, W. M. P. (2011). Process mining: discovering and improving spaghetti and lasagna processes. In *Proceedings of the IEEE symposium on computational intelligence and data mining (CIDM)* (pp. 1–7). IEEE.
van der Aalst, W. M. P. (2016). *Process mining: Data science in action*. Springer.
Abellán, J., & Moral, S. (2003). Building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems, 18*(12), 1215–1225.
Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., et al. (2016). mlr: Machine learning in R. *Journal of Machine Learning Research, 17*(170), 1–5.
Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.
Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.
Breuker, D., Delfmann, P., Matzner, M., & Becker, J. (2014). Designing and evaluating an interpretable predictive modeling technique for business processes. In *Proceedings of the international conference on business process management* (pp. 541–553). Springer.
Cabanillas, C., Di Ciccio, C., Mendling, J., & Baumgrass, A. (2014). Predictive task monitoring for business processes. In *Proceedings of the international conference on business process management* (pp. 424–432). Springer.
Candel, A., Parmar, V., LeDell, E., & Arora, A. (2016). *Deep learning with H_2O*. H_2O.ai Inc.
Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST), 2*(3), 27.
Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the international conference on machine learning, 1995* (pp. 115–123). Elsevier.
Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W. M., & ter Hofstede, A. H. (2015). A recommendation system for predicting risks across multiple business process instances. *Decision Support Systems, 69*, 1–19.
Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297.
Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*(Jan), 1–30.
Di Francescomarino, C., Dumas, M., Maggi, F. M., & Teinemaa, I. (2016). Clustering-based predictive process monitoring. *IEEE Transactions on Services Computing*. doi:10.1109/TSC.2016.2645153.
Di Francescomarino, C., Ghidini, C., Maggi, F. M., & Milani, F. (2018). Predictive process monitoring methods: Which one suits me best? In *Proc. International Conference of Business Process Management* (pp. 462–479).
Dong, L., Frank, E., & Kramer, S. (2005). Ensembles of balanced nested dichotomies for multi-class problems. In *Proceedings of the European conference on principles of data mining and knowledge discovery* (pp. 84–95). Springer.
Evermann, J., Rehse, J.-R., & Fettke, P. (2017). Predicting process behaviour using deep learning. *Decision Support Systems, 100*, 129–140.
Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research, 9*(Aug), 1871–1874.
Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research, 15*(1), 3133–3181.
Frank, E., & Kramer, S. (2004). Ensembles of nested dichotomies for multi-class problems. In *Proceedings of the twenty-first international conference on machine learning* (p. 39). ACM.
Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the international conference on machine learning, ICML: 96* (pp. 148–156). Bari, Italy.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning. *Springer series in statistics*: 1. New York: Springer.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 21*(5), 1189–1232.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*(1), 86–92.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10–18.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832–844.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Holland, B. S., & Copenhaver, M. D. (1987). An improved sequentially rejective Bonferroni test procedure. *Biometrics, 43*(2), 417–423.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning, 11*(1), 63–90.

Iba, W., & Langley, P. (1992). Induction of one-level decision trees. In *Proceedings of the international conference on machine learning,* (pp. 233–240). Elsevier.

IEEE (2016). IEEE 1849–2016 – IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. (pp. 1–50). doi:10.1109/IEEESTD.2016.7740858.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods, 9*(6), 571–595.

Janssenswillen, G., & Depaire, B. (2017). BupaR: Business process analysis in R. In *Proceedings of the fifteenth international conference on business process management (BPM 2017)* (pp. 1–5).

Japkowicz, N., & Shah, M. (2011). *Evaluating learning algorithms: A classification perspective*. Cambridge University Press.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the eleventh conference on uncertainty in artificial intelligence* (pp. 338–345). Morgan Kaufmann Publishers Inc.

Kang, B., Kim, D., & Kang, S.-H. (2012). Periodic performance prediction for real-time business process monitoring. *Industrial Management & Data Systems, 112*(1), 4–23.

Karpathy, A. (2015). The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Online; accessed 10-February-2019].

Leitner, P., Ferner, J., Hummer, W., & Dustdar, S. (2013). Data-driven and automated prediction of service level agreement violations in service compositions. *Distributed and Parallel Databases, 31*(3), 447–470.

Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., & Maggi, F. M. (2015). Complex symbolic sequence encodings for predictive monitoring of business processes. In *Proceedings of the international conference on business process management* (pp. 297–313). Springer.

Macia, N., & Bernadó-Mansilla, E. (2014). Towards UCI+: A mindful repository design. *Information Sciences, 261*, 237–262.

Maggi, F. M., Di Francescomarino, C., Dumas, M., & Ghidini, C. (2014). Predictive monitoring of business processes. In *Proceedings of the international conference on advanced information systems engineering* (pp. 457–472). Springer.

Mannhardt, F., & Blinde, D. (2017). Analyzing the trajectories of patients with sepsis using process mining. *RADAR+ EMISA, 1859*, 72–80.

Mannhardt, F., De Leoni, M., Reijers, H. A., & Van Der Aalst, W. M. (2016). Balanced multi-perspective checking of process conformance. *Computing, 98*(4), 407–437.

Mannhardt, F., de Leoni, M., Reijers, H. A., & van der Aalst, W. M. (2017). Data-driven process discovery-revealing conditional infrequent behavior from event logs. In *Proceedings of the international conference on advanced information systems engineering* (pp. 545–560). Springer.

Márquez-Chamorro, A. E., Resinas, M., & Ruiz-Cortés, A. (2017). Predictive monitoring of business processes: A survey. *IEEE Transactions on Services Computing, 11*(6), 962–977.

Márquez-Chamorro, A. E., Resinas, M., Ruiz-Cortés, A., & Toro, M. (2017). Run-time prediction of business process indicators using evolutionary decision rules. *Expert Systems with Applications, 87*, 1–14.

Melville, P., & Mooney, R. J. (2003). Constructing diverse classifier ensembles using artificial training examples. In *Proceedings of the international joint conferences on artificial intelligence, IJCAI: 3* (pp. 505–510).

Melville, P., & Mooney, R. J. (2005). Creating diversity in ensembles using artificial data. *Information Fusion, 6*(1), 99–111.

Nemenyi, P. (1962). Distribution-free multiple comparisons. *Biometrics, 18*(2), 263.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann Publisher.

Rom, D. M. (1990). A sequentially rejective test procedure based on a modified Bonferroni inequality. *Biometrika, 77*(3), 663–665.

Santoso, A. (2018). Specification-driven multi-perspective predictive business process monitoring. In *Proceedings of the international conference on enterprise, business-process and information systems modeling* (pp. 97–113). Springer.

Tax, N., Verenich, I., La Rosa, M., & Dumas, M. (2017). Predictive business process monitoring with LSTM neural networks. In *Proceedings of the international conference on advanced information systems engineering* (pp. 477–492). Springer.

Teinemaa, I., Dumas, M., La Rosa, M., & Maggi, F. M. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data, 13*(2) article no 17.

Teinemaa, I., Dumas, M., Maggi, F. M., & Di Francescomarino, C. (2016). Predictive business process monitoring with structured and unstructured data. In *Proceedings of the international conference on business process management* (pp. 401–417). Springer.

Ting, K. M., & Witten, I. H. (1997). Stacking bagged and dagged models. In D. H. Fisher (Ed.), *Proceedings of the fourteenth international conference on machine learning* (pp. 367–375). San Francisco, CA: Morgan Kaufmann Publishers.

Unuvar, M., Lakshmanan, G. T., & Doganata, Y. N. (2016). Leveraging path information to generate predictions for parallel business processes. *Knowledge and Information Systems, 47*(2), 433–461.

Verenich, I., Dumas, M., La Rosa, M., Maggi, F., & Teinemaa, I. (2018). arXiv preprint: 1805.02896.

Verenich, I., Dumas, M., La Rosa, M., Maggi, F. M., & Di Francescomarino, C. (2016). Minimizing overprocessing waste in business processes via predictive activity ordering. In *Proceedings of the international conference on advanced information systems engineering* (pp. 186–202). Springer.

Webb, G. I. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning, 40*(2), 159–196.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation, 8*(7), 1341–1390.

Wright, S. P. (1992). Adjusted p-values for simultaneous inference. *Biometrics, 48*(1), 1005–1013.