Int. J. Nonlinear Anal. Appl. 10 (2019) No. 2, 153-166
ISSN: 2008-6822 (electronic)
https://dx.doi.org/10.22075/IJNAA.2019.4182

Investigation of the Effect of Concept Drift on Data-Aware Remaining Time Prediction of Business Processes

Iman Firouzian*, Morteza Zahedi, Hamid Hassanpour

Faculty of Computer Engineering and IT, Shahrood University of Technology, Shahrood, Iran

(Communicated by M.B. Ghaemi)

Abstract

Process Mining is a rather new research area in artificial intelligence with handles event logs usually recorded by information systems. Although, remaining time prediction of ongoing business instances has been always a research question in this area, most of the existing literature does not take into account dynamicity of environment and the underlying process commonly known as concept drift. In this paper, a two-phase approach is presented to predict the remaining time of ongoing process instances; in the first phase, future path of process instances is predicted using an annotated transition system with Fuzzy Support Vector Machine probabilities based on case data and in the second phase, the remaining time is predicted by summing up the duration of future activities each estimated by Support Vector Regressor. Finally, a concept drift adaptation method is proposed. To benchmark the proposed prediction method along with the proposed concept drift adaptation method, experiments are conducted using a real-world event log and a simulation event log. The results show that the proposed approach gained 13% improvement on remaining time prediction in case of concept drift.

Keywords: Business Process, Process Mining, Remaining Time Prediction, Concept Drift.
2010 MSC: 68N30

1. Introduction

An increasing number of companies are using Process Aware Information Systems (PAIS) to support their business. All these systems record execution traces, called event logs, with information

*Iman Firouzian
Email address: iman.firouzian@shahroodut.ac.ir, zahedi@shahroodut.ac.ir, h.hassanpour@shahroodut.ac.ir (Iman Firouzian*, Morteza Zahedi, Hamid Hassanpour)

Received: Jan 2018 *Revised:* Jan 2019

related to executed activities. In typical scenarios, these traces do not only contain which activities have been performed, but also additional attributes. One of the most challenging tasks in process mining is prediction. It is clear that predictions are useful only when the prediction items have not been observed yet. For this reason, remaining time prediction is performed on incomplete traces, at runtime. Nevertheless, the ability to predict the future of a running case is difcult also due to drifts occurring in the model.

Concept drift is the phenomenon where relations between input (features) and output (target variables) change over time, because the underlying models and/or distributions change [1]. A concept drift in business processes is dened as a statistically signicant change in the process behavior [2, 3]. Concept drift effects may occur from three main different perspectives in the context of business processes including the control-ow, data and resource perspectives. Concept drift also represents four different behaviors over time: sudden drift, gradual drift, recurring drift, incremental drift.

In this paper, a remaining time prediction method along with a concept adaptation method is presented. In the proposed remaining time prediction algorithm, a transition system is constructed and annotated with probabilities obtained from Fuzzy Support Vector Machines based on process instance data. Future activities of process instances is obtained using a shortest path algorithm on the annotated transition system. Thereafter, the duration of each of the future activities are estimated using Support Vector Machines. The predicted remaining time is obtained by summing up the durations of future activities.

To make the proposed time prediction algorithm resistant to dynamicity of underlying model commonly known as concept drift, a concept adaptation is proposed solution which uses multiple machine learning models trained over different intervals of previous data, and assigns weights to the predictions of each model based on the difference in time between the model and the test data, factoring in an exponential decay (giving priority to recent models) and a periodic function (giving priority to seasonally similar points in time).

The rest of the paper is organized as follows. In section 2 some related literature are discussed. In section 3, two main techniques of the proposed approach including transition system and Fuzzy Support Vector Machines are technically explained. Section 4 proposes an approach for remaining time prediction along with concept adaptation. Section 5 describes the experimental results for the simulation and the real-world dataset. Section 6 concludes the paper and provides suggestions for future research.

2. Related Works

The first framework which focused on the time perspective has been proposed by van der Aalst et al. [4]. They describe a prediction approach which extracts a transition system from the event log and decorates it with time information extracted from historical cases. The transition system consists in a finite state machine built with respect to a given abstraction of the events inside the event log. The time prediction is made using the time information (e.g., mean duration of specific class of cases) inside the state of the transition system corresponding to the current running instance. A closely related approach is presented in [5], where the same idea of a decorated (a.k.a. annotated) transition system is used. In this approach the annotations are decision trees used to predict the completion time and also the next future activity of the process instance. The work presented in [6] considers the data perspective in order to identify SLAs (Service Level Agreement) violations. In this work, authors try to estimate the amount of unknown data, in order to improve the quality of the final prediction. The actual forecast is built using a multilayer perceptron, trained with the back propagation algorithm. Two works by Folino et al. [7, 8] report an extended version of the technique

described in [4]. In particular, they cluster the log traces according to the corresponding "context features" and then, for each cluster, they create a predictive model using the same method as in [4]. The clustering phase is performed using predictive clustering trees. In order to propose a forecast, the approach clusters the new running instance and then uses the model belonging to the specific cluster. One of the weaknesses of these methods based on [4] is that they assume a static process, where the event log used for the training phase contains all the possible process behaviors. Unfortunately, in real life cases this assumption is usually not valid. Lakshmanan et al. [9] reports an approach which uses the Instance-specific Probabilistic Process Models (PPM) and is able to predict the likelihood of future activities. Even though the method does not provide a time prediction, it gives to the business managers useful information regarding the progress of the process. This work also shows that the PPM built is actually Markovian. A probabilistic method, based on Hidden Markov Model (HMM), is also proposed in [10]. Results show how the proposed method outperforms the state-of-the-art. However, the achieved results seem to be not much different from the ones achieved by [4]. Ghattas et al., in a recent work [11], exploit Generic Process Model and decision trees, based on the process context, to provide decision criteria defined according to the actual process goals. In [12], de Leoni et al. propose a general framework able to find correlation between business process characteristics. They manipulate the event log in order to enrich it with derived information, and then generate a decision tree in order to discover correlations. In particular, one of the correlation problem suggested here is the forecast of the remaining time of running cases. Being based on decision trees, numeric values need to be discretized and this lower the accuracy of the method. For this reason, they do not provide any prediction example. In [13], Polato et al. show an approach based on [4] in which the additional attributes of the events are taken into account in order to refine the prediction quality. This method exploits the idea of annotating a transition system adding machine learning models, such as Naïve Bayes and Support Vector Regressors. The experimental results show how the additional attributes can influence positively the prediction quality.
In this paper we propose an approach based on Fuzzy Support Vector Machine and Support Vector Regression and we discuss their strengths and their weaknesses compared with the approaches presented in [14] and [4]. In particular we emphasize in which scenarios an approach is better than the others and why. A similar approach is presented in [5], in which authors create a process model similar to a transition system, in which non frequent behaviors of the process are discarded. Predictions are made on top of this model using decision trees. This method is very similar to the one presented in [13], the main difference is the process model which is a kind of reduced transition system. More recently, deep neural network based approaches have been proposed. In [15], authors present a recurrent neural network (RNN) for the prediction of the next event, however they do not predict the remaining time. Their network is composed by two hidden RNN layers using basic LSTM cells (Long-short term memory). To train the model, GPUs have been used because of the size of the deep network. Another recent approach based on LSTM is the one by Tax et al. [14]. In that work authors propose a LSTM neural network architecture for predicting the next activity and its execution time, from a partial trace. Such a network can be adopted for remaining time prediction by iteratively predicting the next activity and its duration, until a special end of case activity is predicted. Both the cited deep NN based methods do not take into account additional attributes. A known problem of the deep NN based approaches is their training time and their sensitivity to the hyper parameters choice [16].
Approaches coming from different area scan also be used to achieve similar results. For example, queue theory [17], [18] and queue mining can be seen as a very specific type of process mining, and recent works are starting to aim for similar prediction purposes [19]. In this particular case, authors decided to focus on the delay prediction problem (i.e., providing information to user waiting in lines,

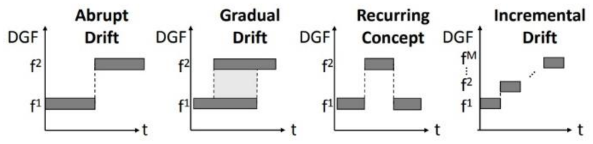

Figure 1: Types of concept drift

in order to improve customer satisfactions). The method presented here is based on the construction of an annotated transition system. Such model is then used to make delay predictions using simple averages or non-linear regression algorithms. They also show other methods based on queue mining techniques. Even though this approach concerns making predictions on the time perspective, the slant is totally different and the goal is narrower with respect to the aim of the approach we propose in this work. Another example of prediction-focused paper has recently been published by Rogge-Solti and Weske [20]. In this case the focus is on the prediction of the remaining time, in order to avoid missing deadlines. However, only information about the workflow are used to build the model and to make prediction.
Remaining time prediction is a particular instantiation of the so-called predictive monitoring task, in which the goal is to provide early advice so that users can steer ongoing process executions towards the achievement of business constraints. Maggi et al. [21] proposed a decision tree based predictor model. The presented framework continuously estimate how likely is that a user defined constraint it will be fulfilled by the ongoing process instances. The same task is faced in [22]. Here authors present an alternative approach where traces are treated as complex symbolic sequences. They propose two possible encodings: (i) index-based and (ii) a combination of the first one and an HMM based one. From an Area Under the Roc Curve perspective the proposed encoding outperforms the simple baselines (i.e., boolean, frequency-based). Using the just mentioned encoding, Verenich et al. [23] proposes a two phase approach: in the first phase the dataset is divided into groups by means of unsupervised clustering; in the second phase a predictor is trained for each cluster. In particular they use random forests as the predictor algorithm. A very similar approach is presented in [23]. Finally, Teinemaa et al. [24] show how textual information, when sufficiently large, can be useful to improve accuracy and earliness. Even in this work the best performing predictors are random forests.
While concept drift is often neglected in process mining, a notable discussion of this topic has been presented by Bose et al. [25]. The authors raise the awareness for concept drift relating to the control-flow, data, and resource dimension of business processes. Here, changes in control-flow are closely related to patterns for behavioral and structural changes in business processes [26]. Bose et al. [25] further highlight the relevance of the general types of concept drift outlined in Figure 1 also for process mining applications. Yet, the solutions provided by [25] relate to the mere detection of abrupt concept drift: based on temporal windows, significant changes in the control-flow of a process are identified. Similar approaches, also with a control-flow focus, have been put forward in [27] to detect abrupt drifts. These ideas have been tailored to detect gradual and recurring drift in [28]. Existing techniques for PPM, however, do not account for drifts in the underlying business processes. A notable exception has been presented by Berti [29] for cycle time prediction. Since the work assumes a-priori knowledge of the concept drift points, it is not generally applicable, though.

3. TECHNIQUES USED IN PROPOSED METHOD

3.1. TRANSITION SYSTEM

Transition system is an abstraction machine which is defined to be in one of possible finite states. By activating a transition, the machine changes from one state to the other, but the machine can be in only one state at the time. Transition system can also be represented with a directed graph, in which every node represents a state and every edge represents a transition from one state to the other. The transition system is defined in formal as follows:

Definition 1 (TS transition system). A transition system is a triple TS =(S, A, T) in which S is a set of possible states, E is a set of possible events (labeled transitions) and $T \subseteq S \times E \times S$ is a set of transitions which describes a movement of the system from one state to the other. $S^{start} \subseteq S$ is a set of starting states and $S^{start} \subseteq S$ is a set of ending states. Transition system is constructed by state representation and event representation functions which is also called as abstractions.

Definition 2 (State Representation Function). Let R$_S$ be a set of state representations. A state representation function $f^{state} \in \sum \rightarrow R_S$ is a function that given a partial trace σ, the corresponding representation is returned (e.g. sequences, sets, multiset over some event properties) [4].

Here, a trace is the same as a completed business process, while a partial trace represents a running business process instance. According to research [30], three common abstractions are as follows:

1. Sequence abstraction which represents only the order of activities in each state.
2. Multiset abstraction which represents number of execution of each activity regardless of the order.
3. Set abstraction represents pure existence of activities regardless of the order and number of executions of each.

Definition 3 (Event Representation Function). Let R$_e$ be a set of event representations. An event representation function $f^{event} \in E \rightarrow R_e$ is a function that given an event e returns a representation for it (projection function over $e \in E$). In the proposed approach, a transition system which leverages f^{state} and f^{event} functions and maps states of prefixes of partial traces to event representations using f^{event} function. In Figure 1, an event log is sample event log is shown with the extracted transition system. Each state s$_0$, ... , s$_5$ is labeled with a state representation function and each transition is labeled with an event representation function.

3.2. FUZZY SUPPORT VECTOR MACHINE

In standard SVM classifer, the significance of error rate (value of ϵ_i variables) for all training samples is set as equal, while it is in some case necessary to modify this assumption of the standard SVM. Using Fuzzy logic, one could incorporate the significance of each sample in training phase. It is also possible to choose soft decision over hard decision using Fuzzy logic. In the next section, we explain the details of the underlying theory. Note that the core of Fuzzy-SVM is explained while it is supposed the reader is familiar with the basics.

3.3. INCORPORATING SIGNIFICANCE OF SAMPLES

Standard SVM treats training samples as pairs of (x$_i$, y$_i$), where y$_i\epsilon${-1,+1}. To incorporate the significance of samples, the samples are considered as triples (x$_i$, y$_i$, s$_i$). s$_i$ actually represents a membership degree of x$_i$ to the associated class. SVM formulation is modified to include the membership degree as follows:

$$\begin{cases} Minimize_w & \frac{1}{2}\, w.w + C\sum_{i=1}^{N} s_i\varepsilon_i \\ subject\ to & y_i\,(w.x_i + b) \geq 1 -\ \varepsilon_i \\ \qquad \varepsilon_i \geq 0,\ for\ i = 1,\ \ldots,\ N & \end{cases} \tag{3.1}$$

Searching the optimal hyperplane in (3.1) is a QP problem, which can be solved by constructing a Lagrangian and transformed into the dual

$$\begin{cases} Maximize_{\alpha} & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)+\sum_{i=1}^{N}\alpha_i \\ subject\ to & \sum_{i=1}^{N}\alpha_i y_i = 0,\ 0\ \le\ \alpha_i\ \le\ s_i C \\ & ,\ for\ i=1,\ \ldots,\ N \end{cases} \tag{3.2}$$

In fact, the difference between standard SVM and Fuzzy SVM is that upper bound of α_i lagrange multiplier is equal to s$_i$C, while this bound is equal to C in standard SVM. Support vectors are patterns for which the corresponding lagrange multiplier is $0 < \alpha_i \le$ s$_i$C. A set of support vectors which satisfy the condition $0 < \alpha_i <$ s$_i$C, is used in computations of b.

Now, it's time to define membership degree s$_i$ of each sample x$_i$. According to[31] and[32] , s$_i$ is computed based on the ratio of the distance between each sample and center of the class to the distance between the furthest sample of the class and center of the class. This approach is sensitive to outlier data and it is not recommended where outlier data exist. s$_i$ is computed as follows:

$$s_i = \exp\left(-\frac{1}{2}(x_i-\mu)\right)^T \sum{}^{-1}(x_i-\mu)) + \varepsilon \tag{3.3}$$

Here, ϵ is a small number with value set to 0.01. μ and σ are average vector and covariance matrix associated to class of sample x$_i$, respectively. For simplicity and complexity reduction, one could consider covariance matrix as diagonal matrix. To reduce the effect of outlier data in SVM training phase, the significance level of each sample (s$_i$) is multiplied by error of the sample (ϵ_i).

3.4. SOFT DECISION USING FUZZY LOGIC

In this paper, proposed approach is targeted to the problem of multi-class classification which has a pairwise solution. If weight vector and threshold level for separation of classes i and j be w$_{ij}$ and b$_{ij,}$ accordingly, then decision function for separation of classes i and j would be as follows:

$$D_{ij}\ (x) = w_{ij}^T\ x +\ b_{ij} \quad , \quad i \ne j \tag{3.4}$$

$$D_{ji}\ (x) =\ -D_{ij}(x)$$

In standard SVM, decision function for recognition of class i samples against other classes is as follows:

$$D_i\ (x) = \sum_{j=1, j\ne i}^{C} sign(D_{ij}(x)) \tag{3.5}$$

where c is a number of classes. Class of vector x is defined as follows:

$$c\,(x) = \underset{i}{\operatorname{argmax}}\, D_i(x) \tag{3.6}$$

Even though all classes cover the classification region, there would still be some regions called as unclassifiable regions. A solution to the problem is using soft decision in fuzzy logic. A membership function m$_{ij}$(x) is defined in order to distinct between classes i and j. Function m$_{ij}$(x) represents the assignment level of vector x to class i; in other words, it shows how much class i is winner and how much class j is loser:

$$m_{ij}\,(x) = \frac{1}{1+\exp(-D_{ij}(x))} \tag{3.7}$$

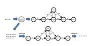

Figure 2: Annotated transition system enriched with Fuzzy-SVM membership degrees

To assign vector x to class i in all class pairs (i, j) and let class i be winner in all pairwise decisions, Fuzzy AND operator and Min operator is then used:

$$m_i\left(x\right) = \min_{j, \quad j \neq i} m_{ij}(x) \tag{3.8}$$

Having computed m$_i$(x), vector x is assigned to class c(x):

$$c\left(x\right) = \operatorname*{argmax}_{i} m_i(x) \tag{3.9}$$

Using Fuzzy logic and soft decision, not only unclassified regions problem is solved, but also classification is done using soft Fuzzy operations.

4. PROPOSED APPROACH

In the proposed approach, a method for predicting the remaining time of business processes is predicted in the first subsection and a method for concept drift adaptation method in the context of business process remaining time prediction is presented in the second subsection.

4.1. PREDICTION OF REMAINING TIME

Given an event log containing completed business processes and a partial trace of a running business process instance, the aim is to first predict the future path and then predict the remaining time of process instances. Our approach is an improvement to the work presented by Polato et al. [13] which leverages naïve bayes classifier to estimate the probability of the transition from current state to next state. One of the assumptions in [13] is the conditional independence of predictors, which is actually the independence of case data attributes in different activities. Since in real-world business process scenarios, data attributes are usually dependent, in the proposed approach, sequence of activities and case data attributes is used in Fuzzy-SVM classification for future patch detection in each branching.

Transition system is first extracted from event log using the algorithm presented by Aalst et al. [4]. In the construction of the transition system, sequence abstraction is used in which order of activities is important. Then, the probability of activation of each transition is computed using multi-class Fuzzy-SVM based on previous traces in the event log. Given transition probabilities, the most probable sequence from current state to final state is obtained.

Given a transition system annotated with the transition probabilities, one could find a probability of any sequence of activities S = <s$_1$, s$_2$, ... , s$_n$> by taking the product of transition probabilities between every pair of connecting states:

$$P_S = \prod_{i=1}^{n-1} p_i \tag{4.1}$$

At runtime, we map a partial trace of an ongoing instance to the current state in the transition system. To find the most likely future sequence from its current state until its completion, we use the idea by Polato et al. [13], who transform the problem of finding the sequence with the highest probability into the shortest path problem. Indeed, an annotated transition system can be viewed as a weighted directed graph, where nodes are states, edges are possible transitions, and weights are transition probabilities. Consequently, we would like to find the shortest path between a current node and the target node, which corresponds to the final state of the process. However, in the shortest path problem, the sum of edge weights, rather than the product, is to be minimized. Thus, instead of using transition probabilities as the edge weights, we use their logarithm. Furthermore, to ensure that the transitions with low probability correspond to edges with high weights and vice versa, we take the absolute value of the logarithm of transition probability. Then we have to minimize the sum:

$$\sum_{i=1}^{n-1} |log\ p_i| \tag{4.2}$$

It should be noted that in the shortest path problem we have to explicitly specify current node and target node. To account for the fact that processes may have multiple final states, we add a single "exit" state, and connect each possible final state to the "exit". Then the transition probability from each final state to the "exit" will be the probability of an instance ending with this state.

Finally, the predicted remaining time is obtained by summing up the durations of future activities, each estimated by Support Vector Regressor.

4.2. Incorporating the Concept Drift

Let D be our dataset, and the points in D be d$_i$. Each data point has input features d$_i$.x, a timestamp d$_i$.time, and a label d$_i$.y. In our use case of business process remaining time prediction, d$_i$ is a case, d$_i$.x is information about a case, d$_i$.time is the timestamp of the first event of a case, and d$_i$.y is the predicted remaining time as labeled. D contains data over a time period of length $T = max_{d_i \in D} d_i.time - \ min_{d_i \in D} d_i.time$. This length is divided into smaller datasets D$_j$, each with length S, such that $d_i \in D_j \iff \lfloor \frac{d_i.time}{S} \rfloor = j$. In this paper S is regarded as a constant in time, and different values of S are explored. Picking the right value of S is important, having too short time-frames will mean too few data points, having too long time-frames might fail to capture short-term concept drift. The splitting of data is visualized in Figure 3. We then train a machine learning model on each D$_j$, resulting in multiple classifiers M$_j$. This creates an ensemble classifier, and we use a weighting function explained below to decide the importance of each classifier in the ensemble. In the running example, one could decide to train a classifier monthly (S = 1month), creating twelve models per year, each consisting of all business process instances that instantiated in a specific month.

When predicting a test data point, all models M$_j$ receive a weight based on how old the model is. We propose a combination of two weights. The first weight is related to exponential decay, and is inspired by for example [1]. Let d be a test data point, which has a timestamp d.time. As such, d belongs to period $j = \lfloor \frac{d.time}{S} \rfloor$. We assign weight 1 to M$_{j-1}$,$10^{-\beta}$ to M$_{j-2}$,$10^{-2\beta}$ to M$_{j-3}$ and soon; with $\beta \in$ IR+. In general we assign $10^{-(j-k+1)\beta}$ to model M$_k$, for k < j (models M$_k$, k $\geq$ j are not available for testing d, since they are learned from data that is not available at d.time).

The second weight relates to giving extra importance to models that are seasonally similar. Say we want to reduce the effects of recurrent concept drift that occurs every p $\in$ IR+ time. We hence add weight $\tau \in$ IR+ to models that belong to d.time−p, d.time−2p and so on. These additions are given on top of the exponential decay weight; though in most situations we will have that p >> S (we train models far more often than the recurrent concept drift occurs), and hence the exponential weight will be insignificant for models around time d.time−p. The determination of the weights is also presented in Algorithm1.

We combine the models as follows: for data points in D$_j$ we let each of the j existing trained prediction models (M$_0$,M$_1$,...,M$_{j-1}$) make a prediction, resulting in j vectors (p$_{1,k}$,p$_{2,k}$,...,p$_{n,k}$), where

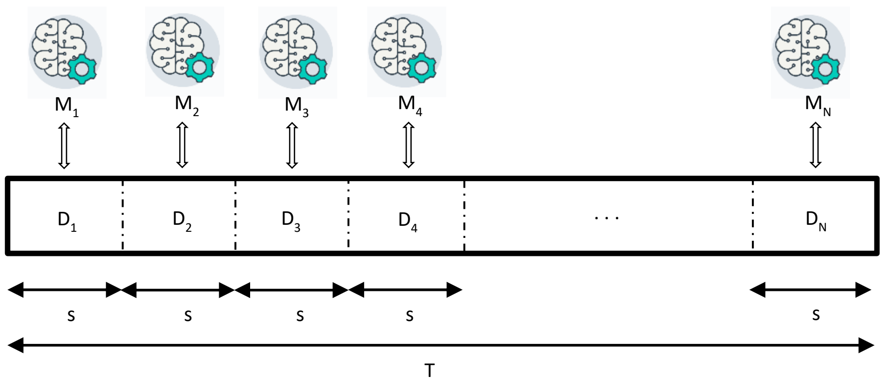

Figure 3: Concept drift adaptation using event log splitting

$p_{l,k}$ is the probability of label l as predicted by model k (where we have a total of n labels in all of D). We determine the combined prediction for the test data point d by taking the weighted sum:

$$p_l = \sum_{0\ \leq k\ \leq j-1} w_k . p_{l,\ k} \tag{4.3}$$

where w_k is the weight of the described above. The predicted label is then arg $\max_l$ p_l. The combination of β, τ and p uniquely identifies a weighting assignment, together with S we form a complete concept drift adaption technique. This technique adapts to gradual and recurrent drift. Finding the best combination of β,τ,p and S is part of the training process. In this paper we will restrict to a single value for p.

```
$Algorithm 1 : ensemble\ classifier\ weighting$
  $j \leftarrow \left\lfloor \frac{d.time}{S} \right\rfloor$
  $for\ k = 0 \ldots j-1\ do$
    $w_k \leftarrow 10^{-(j-k+1)\beta}$
  $end$
  $t \leftarrow d.time - p$
  $while\ t > 0\ do$
    $k \leftarrow \left\lfloor \frac{t}{S} \right\rfloor$
    $w_k \leftarrow w_k + \tau$
    $t \leftarrow t - p$
  $end$
```

5. EXPERIMENTAL RESULTS

In this section we show the results of applying our approach to the simulation dataset and the real-world dataset. Since the proposed solution aims to adapt to overcome the effects of concept drift

Table 1: F1 and accuracy scores for all configurations of β, τ and S over simulation study

S	τ	$\beta = 0$		$\beta = 1$		$\beta = 2$	
		ACC	F_1	ACC	F_1	ACC	F_1
1	0	0.14	0.109	0.669	0.672	0.667	0.671
	0.01	0.139	0.108	0.670	0.673	0.669	0.673
	0.1	0.154	0.130	0.671	0.674	0.670	0.674
	1	0.263	0.295	0.742	0.653	0.742	0.652
	10	0.649	0.605	0.723	0.642	0.722	0.642
	100	0.709	0.635	0.716	0.639	0.716	0.639
2	0	0.113	0.090	0.344	0.348	0.395	0.396
	0.01	0.114	0.092	0.352	0.355	0.422	0.444
	0.1	0.134	0.118	0.513	0.516	0.528	0.532
	1	0.245	0.279	0.607	0.549	0.605	0.545
	10	0.547	0.515	0.582	0.529	0.582	0.530
	100	0.575	0.526	0.577	0.527	0.577	0.527
3	0	0.123	0.090	0.161	0.171	0.161	0.171
	0.01	0.127	0.096	0.161	0.171	0.161	0.171
	0.1	0.142	0.119	0.162	0.172	0.162	0.172
	1	0.381	0.426	0.731	0.649	0.731	0.648
	10	0.724	0.641	0.734	0.646	0.734	0.646
	100	0.731	0.645	0.734	0.646	0.734	0.646
4	0	0.096	0.075	0.180	0.185	0.186	0.192
	0.01	0.097	0.077	0.185	0.190	0.188	0.194
	0.1	0.112	0.099	0.203	0.207	0.204	0.209
	1	0.215	0.241	0.454	0.430	0.454	0.428
	10	0.450	0.415	0.460	0.420	0.460	0.420
	100	0.461	0.422	0.461	0.422	0.461	0.422

and not to detect it, we are only interested in the score, but not in the drift in the process or derived models themselves. We first present and discuss the results of the simulation dataset, and then we do the same for the real-world dataset of BPI challenge 2012.

The results of the simulation study are presented in Figure 4 and Table 1. Figure 4 shows the scores of each of the twelve methods applied. The best scores for the proposed method are underlined in Table 1 to indicate the optimal values of β and τ. Table 1 further shows the F1 and accuracy scores for all configurations of β, τ and S.

The results of the Simulation in terms of F1 scores are presented in Figure 1. Three methods including the proposed method, first model and last model are tested. First model is a model created once on the first year of the dataset. Last model uses the most recent model. Each method is tested for different values of S, indicated by the numbers. The values of β and τ for the proposed method are underlined in Table 1.

The results of the real-world dataset of BPI Challenge 2012 are presented in Figure 5 and Table 2. Figure 5 shows the overall scores of each of the twelve methods applied, and Table 3 presents the values of β and τ that belong to each value of S for the proposed method.

The key result to take away from Figure 5 is that the improvement of the proposed method over last model is present, albeit small. The most likely explanation is that there is no strong effect of recurrent concept drift present in the real-world dataset. This also matches the results for τ; only S = 28 has a value for τ that puts significant weight to periodically similar models, though for S = 28, there will be two such models at best. Please note that the method: Last model does not exist, and that we used it as an intuitive method for comparison. The results do not disprove the effectiveness

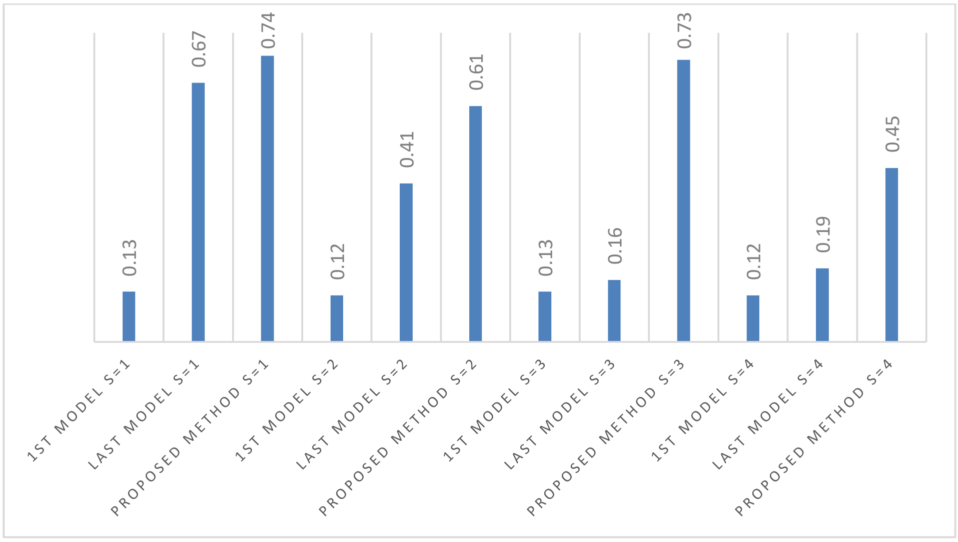

Figure 4: F1 scores of three different methods (Proposed method, First Model, Last Model) over simulation study for four different values of S

Table 2: F1 and accuracy scores for all configurations of β, τ and S over real-world dataset

Proposed Method					Last Model		First Model	
7	1	0	0.417	0.418	0.405	0.405	0.251	0.245
14	1	0.1	0.418	0.417	0.412	0.413	0.253	0.245
21	1	0	0.412	0.410	0.405	0.406	0.264	0.258
28	1	1	0.419	0.416	0.415	0.418	0.267	0.259

of the proposed method. The goal of the proposed method was to allow adaption to both gradual and recurrent concept drift. As is evident from the results for τ, the proposed method is in essence self-correcting the lack of recurrent concept drift.

The results of the Real-world dataset in terms of accuracy scores. Three methods including the proposed method, first model and last model are tested. First model is a model created once on the first year of the dataset. Last model uses the most recent model. Each method is tested for different values of S, indicated by the numbers. Each method is tested for different values of S, indicated by the numbers.

6. CONCLUSION

In this paper, a remaining time prediction method along with a concept adaptation method is presented. In the proposed remaining time prediction algorithm, a transition system is constructed and annotated with instance-specific probabilities obtained from Fuzzy Support Vector Machines based on process instance data. Future activities of process instances is obtained using a shortest path algorithm on the annotated transition system. Thereafter, the duration of each of the future activities are estimated using Support Vector Machine. The predicted remaining time is obtained by summing up the durations of future activities. To adapt to concept drift conditions, a multi-model is proposed which is trained over different intervals of previous data, and assigns weights to the predictions of each model based on the difference in time between the model and the test data, factoring in an exponential decay and a periodic function. Experimental results show that the concept drift adaptation method gained 13% improvement on remaining time prediction. Future

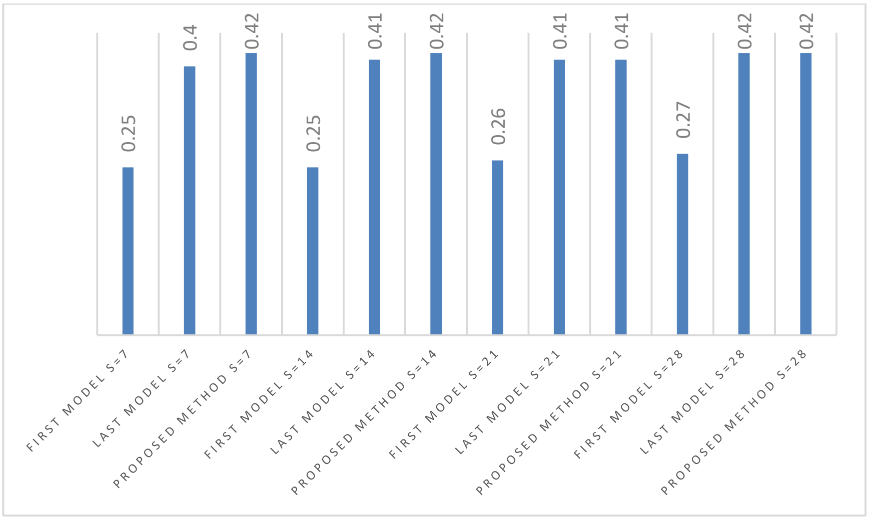

Figure 5: F1 scores of three different methods (Proposed method, First Model, Last Model) over real-world dataset for four different values of S

works requires to more intelligently and dynamically determine the period for each model, and also break the assumption of constant period for all models.

7. References

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM computing surveys (CSUR), 46 (2014) 44.
[2] R.J.C. Bose, W.M. van der Aalst, I. Žliobaitė, M. Pechenizkiy, Handling concept drift in process mining, in: International Conference on Advanced Information Systems Engineering, Springer, 2011, pp. 391-405.
[3] A. Maaradji, M. Dumas, M. La Rosa, A. Ostovar, Fast and accurate business process drift detection, in: International Conference on Business Process Management, Springer, 2016, pp. 406-422.
[4] W.M. Van der Aalst, M.H. Schonenberg, M. Song, Time prediction based on process mining, Information systems, 36 (2011) 450-475.
[5] M. Ceci, P.F. Lanotte, F. Fumarola, D.P. Cavallo, D. Malerba, Completion time and next activity prediction of processes using sequential pattern mining, in: International Conference on Discovery Science, Springer, 2014, pp. 49-61.
[6] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, F. Leymann, Runtime prediction of service level agreement violations for composite services, in: Service-oriented computing. ICSOC/ServiceWave 2009 workshops, Springer, 2009, pp. 176-186.
[7] F. Folino, M. Guarascio, L. Pontieri, Discovering context-aware models for predicting business process performances, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2012, pp. 287-304.
[8] F. Folino, M. Guarascio, L. Pontieri, Discovering high-level performance models for ticket resolution processes, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2013, pp. 275-282.
[9] G.T. Lakshmanan, D. Shamsi, Y.N. Doganata, M. Unuvar, R. Khalaf, A markov prediction model for data-driven semi-structured business processes, Knowledge and Information Systems, 42 (2015) 97-126.
[10] S. Pandey, S. Nepal, S. Chen, A test-bed for the evaluation of business process prediction tech-

niques, in: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), IEEE, 2011, pp. 382-391.

[11] J. Ghattas, P. Soffer, M. Peleg, Improving business process decision making based on past experience, Decision Support Systems, 59 (2014) 93-107.

[12] M. De Leoni, W.M. Van der Aalst, M. Dees, A general framework for correlating business process characteristics, in: International Conference on Business Process Management, Springer, 2014, pp. 250-266.

[13] M. Polato, A. Sperduti, A. Burattin, M. de Leoni, Data-aware remaining time prediction of business process instances, in: 2014 International Joint Conference on Neural Networks (IJCNN), IEEE, 2014, pp. 816-823.

[14] N. Tax, I. Verenich, M. La Rosa, M. Dumas, Predictive business process monitoring with LSTM neural networks, in: International Conference on Advanced Information Systems Engineering, Springer, 2017, pp. 477-492.

[15] J. Evermann, J.-R. Rehse, P. Fettke, A deep learning approach for predicting process behaviour at runtime, in: International Conference on Business Process Management, Springer, 2016, pp. 327-338.

[16] T. Domhan, J.T. Springenberg, F. Hutter, Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[17] G. Bolch, S. Greiner, H. De Meer, K.S. Trivedi, Queueing networks and Markov chains: modeling and performance evaluation with computer science applications, John Wiley & Sons, 2006.

[18] R.W. Hall, Queueing methods: for services and manufacturing, Pearson College Div, 1991.

[19] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining for delay prediction in multi-class service processes, Information Systems, 53 (2015) 278-295.

[20] A. Rogge-Solti, M. Weske, Prediction of business process durations using non-Markovian stochastic Petri nets, Information Systems, 54 (2015) 1-14.

[21] F.M. Maggi, C. Di Francescomarino, M. Dumas, C. Ghidini, Predictive monitoring of business processes, in: International conference on advanced information systems engineering, Springer, 2014, pp. 457-472.

[22] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, F.M. Maggi, Complex symbolic sequence encodings for predictive monitoring of business processes, in: International Conference on Business Process Management, Springer, 2016, pp. 297-313.

[23] I. Verenich, M. Dumas, M. La Rosa, F.M. Maggi, C. Di Francescomarino, Complex symbolic sequence clustering and multiple classifiers for predictive process monitoring, in: International Conference on Business Process Management, Springer, 2016, pp. 218-229.

[24] I. Teinemaa, M. Dumas, F.M. Maggi, C. Di Francescomarino, Predictive business process monitoring with structured and unstructured data, in: International Conference on Business Process Management, Springer, 2016, pp. 401-417.

[25] R.J.C. Bose, W.M. Van Der Aalst, I. Žliobaitė, M. Pechenizkiy, Dealing with concept drifts in process mining, IEEE transactions on neural networks and learning systems, 25 (2013) 154-171.

[26] B. Weber, M. Reichert, S. Rinderle-Ma, Change patterns and change support features–enhancing flexibility in process-aware information systems, Data & knowledge engineering, 66 (2008) 438-466.

[27] J. Carmona, R. Gavalda, Online techniques for dealing with concept drift in process mining, in: International Symposium on Intelligent Data Analysis, Springer, 2012, pp. 90-102.

[28] J. Martjushev, R.J.C. Bose, W.M. van der Aalst, Change point detection and dealing with gradual and multi-order dynamics in process mining, in: International Conference on Business Informatics Research, Springer, 2015, pp. 161-178.

[29] A. Berti, Improving Process Mining prediction results in processes that change over time, DATA ANALYTICS 2016, (2016) 49.
[30] W.M. Van der Aalst, V. Rubin, H. Verbeek, B.F. van Dongen, E. Kindler, C.W. Günther, Process mining: a two-step approach to balance between underfitting and overfitting, Software & Systems Modeling, 9 (2010) 87.
[31] C.-F. Lin, S.-D. Wang, Fuzzy support vector machines, IEEE transactions on neural networks, 13 (2002) 464-471.
[32] Y. Liu, H. Huang, Fuzzy support vector machines for pattern recognition and data mining, International journal of fuzzy systems, 4 (2002) 826-835.