Cost-Sensitive Predictive Business Process Monitoring

Martin Käppel[1(✉)], Stefan Jablonski[1], and Stefan Schönig[2]

[1] Institute for Computer Science, University of Bayreuth, Bayreuth, Germany
{martin.kaeppel,stefan.jablonski}@uni-bayreuth.de
[2] University of Regensburg, Regensburg, Germany
stefan.schoenig@ur.de

Abstract. In predictive business process monitoring current and historical process data from event logs is used to predict the evolvement of running process instances. A wide number of machine learning approaches, especially different types of artificial neural networks, are successfully applied for this task. Nevertheless, experimental studies revealed that the resulting predictive models are not able to properly predict non-frequent activities. In this paper we investigate the usefulness of the concept of cost-sensitive learning, which introduces a cost model for different activities to better represent them in the training phase. An evaluation of this concept applied to common predictive monitoring approaches on various real life event logs shows encouraging results.

Keywords: Process mining · Predictive business process monitoring · Cost-sensitive learning · Process prediction

1 Introduction

Predictive business process monitoring methods support participants performing a running process instance of a business process [1]. These methods use current and historical process data from event logs to make predictions how a running process instance will evolve up to its completion. These predictions encompass, among other things, performance predictions, predictions regarding the outcome of a process [20], and predictions about upcoming events, including information about activities performed next [4,5,7,16] and by whom [4,17].

Usually, activities (or in general values of event attributes) occur in an event log in a different number. Some activities are executed almost in every process instance, while other activities only occur in a handful of instances. Often, however, such rare activities are of particular importance, since they are only executed in case of handling special or error cases. Usually, such cases bear some challenges (e.g., high costs, lack of experience, increased risk), why it is important to support involved process participants with precise predictions.

In the last years research has shown that deep learning approaches are highly customizable and precise for prediction tasks. However, deeper investigations

L. Bellatreche et al. (Eds.): ADBIS 2021 Short Papers, Workshops and Doctoral Consortium, CCIS 1450, pp. 14–26, 2021.
https://doi.org/10.1007/978-3-030-85082-1_2

reveal that the imbalance of activities (or other event attributes) prevent a precise prediction of non-frequent activities [12,16]. As a result, frequent trace variants (i.e., standard cases) are predicted very well, while rare trace variants are barely predicted correctly, since they are treated in the prediction as a standard case.

Conventional machine learning (ML) algorithms assume an approximately uniform distribution of examples and assume that all prediction errors made by the predictive model are equally weighted [8]. Real-world applications typically have different interpretations for each error. For example, in a helpdesk process with activities for closing, reviewing, and rejecting a ticket, predicting a review of the ticket instead of closing the ticket is less problematic than predicting a rejection of the ticket. Similar problems occur, for instance, in medical treatment processes or processes for claim settlement.

We address the issue of predicting non-frequent trace variants by applying the concept of cost-sensitive learning. Therefore, each class is associated with a given misclassification cost and the used ML algorithm is modified to minimize the total classification costs instead of the number of incorrect predictions.

Next, we provide a introduction in basic terminology and give a formal problem definition. Sect. 3 introduces our approach, while Sect. 5 presents its evaluation. In Sect. 4 we discuss related work. Finally, Sect. 6 outlines future work.

2 Basic Terminology and Problem Statement

2.1 Process Mining and Predictive Business Process Monitoring

The main input of process mining techniques is a *(process) event log*. An event log is a set of traces that are related to a certain business process. A *trace* (also called *case*) is a temporarily ordered sequence of events representing the execution of a process instance. An *event* capsulates information about the execution of an activity (i.e., a step in a business process) within a process instance. These information are characterized by various *event attributes* such as the case id ($\mathcal{C}$), date and time of occurrence ($\mathcal{T}$), the name of the corresponding activity ($\mathcal{A}$), the process participants or systems involved in executing the activity ($\mathcal{L}$), and further data payload ($\mathcal{D}$). Note, that only case identifier, timestamp, and corresponding activity are mandatory, all other event attributes are optional.

Definition 1. *Let $\mathcal{E}$ be the set of all possible event identifiers, $\mathcal{P}$ the set of event attributes, and ε the empty element. For each $p \in \mathcal{P}$, we define a function $\pi_p : \mathcal{E} \rightarrow dom(p) \cup \{\varepsilon\}$ that assigns a value of the domain of p to an event. However, the empty element can only be attached to optional event attributes.*

Let us consider the fragment of an event log reported in Table 1. This event log provides the following event attributes: a *case identifier*, the name of the executed *activity*, the *timestamp* of execution, the involved *resource*, and further information (*amount*, *key*) in form of data payload. For example, for event e_{12}, holds $\pi_{\mathcal{A}}(e_{12})$ = `"T"`, $\pi_{\mathcal{C}}(e_{12})$ = `"Case1"`, $\pi_{\mathcal{L}}(e_{12})$ = `"SL"`, $\pi_{\mathcal{T}}(e_{12})$ = `"2020-10-09T14:51:01"`, $\pi_{\mathcal{D}_{\text{amount}}}(e_{12}) = \varepsilon$, and $\pi_{\mathcal{D}_{\text{key}}}(e_{12})$ = `"HG-4"`.

Table 1. Excerpt of a process event log

Case ID	Event ID	Activity	Timestamp	Resource	Amount	Key
C1	e_{11}	A	2020-10-09T14:50:17	MF		SD-1
C1	e_{12}	T	2020-10-09T14:51:01	SL		HG-4
C1	e_{13}	W	2020-11-09T12:54:39	KH		HZ-2
C2	e_{21}	A	2019-04-03T08:55:38	MF		SD-2
C2	e_{22}	T	2019-04-03T08:55:53	SL	340	HK-7
C2	e_{23}	C	2019-05-19T09:00:28	KH		SGH-3
...	...	...	...	...	...	...

Definition 2. *Let S be the universe of all traces. A* ***trace*** *$\sigma \in S$ is a finite non-empty sequence of events $\sigma = \langle e_1, ..., e_n \rangle$ such that for $1 \leq i < j \leq n : e_i, e_j \in \mathcal{E} \wedge \pi_{\mathcal{C}}(e_i) = \pi_{\mathcal{C}}(e_j) \wedge \pi_{\mathcal{T}}(e_i) \leq \pi_{\mathcal{T}}(e_j)$, where $|\sigma| = n$ denotes the* ***length*** *of σ. A trace σ is called* ***completed*** *if there is no $e' \in \mathcal{E}$ such that $\pi_{\mathcal{C}}(e') = \pi_{\mathcal{C}}(e)$ with $e' \notin \sigma$ and $e \in \sigma$. An* ***event log*** *L is a set $L = \{\sigma_1, ..., \sigma_l\}$ of completed traces.*

Predictive business process monitoring approaches often partition traces for training in sets of prefixes and suffixes to represent uncompleted traces.

Definition 3. *Let $\sigma = \langle e_1, ..., e_n \rangle \in S$ be a trace. We define two functions hd and tl, which return the* ***event prefix*** *of length r (i.e., the first r elements of a trace), respectively the* ***event suffix*** *of length r (i.e., the last r elements of a trace) of σ as follows:*

$$hd : S \times \mathbb{N}_{\geq 0} \to S, \qquad (\sigma, r) \mapsto \begin{cases} \langle\rangle & \text{if } r = 0 \\ \langle e_1, e_2, ..., e_r \rangle & \text{if } 0 < r \leq |\sigma| \\ \sigma & \text{if } r > |\sigma| \end{cases}$$

$$tl : S \times \mathbb{N}_{\geq 0} \to S, \qquad (\sigma, r) \mapsto \begin{cases} \langle\rangle & \text{if } r = 0 \\ \langle e_{r+1}, ..., e_n \rangle & \text{if } 0 < r \leq |\sigma| \\ \sigma & \text{if } r > |\sigma| \end{cases}$$

where $\langle\rangle$ represents the empty trace.

Depending on the prediction target (i.e., event attribute), we can define different prediction functions:

Definition 4. *Let $\sigma = \langle e_1, ..., e_n \rangle \in S$ be an uncompleted trace, $\mathcal{P}$ the set of recorded event attributes, $e' \in \mathcal{E}$ a predicted event, and $r \in \mathbb{N}$, then for each $p \in \mathcal{P}$ the* ***prediction problem*** *is defined as $\Omega_p(hd(\sigma, r)) = \pi_p(e'_{r+1})$.*

For example, the prediction of the next activity is given by $\Omega_{\mathcal{A}}(hd(\sigma, r)) = \pi_{\mathcal{A}}(e_{r+1})$. If we not only want to predict the next event, rather than a whole suffix, we can apply Ω_p recursively over the predictions.

2.2 Formal Problem Definition

According to Definition 4 the prediction of a categorical event attribute $p \in \mathcal{P}$ (e.g., the next activity) is a classification problem γ_k^p with k classes. We assign a class label (i.e., a value of the domain of p) in dependency of the prediction target to a running process instance. We denote the set of class labels for p as $\mathcal{F}_p$. If we consider the class distribution of event attributes in different event logs (or subsets used for training), we always observe an imbalanced class distribution: For some classes (*majority classes*) there are significantly more examples than for other classes (*minority classes*). This observation is typically for the process domain. We denote with $\eta = (\eta_1, \cdots, \eta_k) \in [0,1]^k$ the corresponding probability distribution, where $\eta_i = p(c_i)$ stands for the probability of class c_i. Based on this we define majority and minority classes formally as follows:

Definition 5. *Let γ_k be a classification problem with k classes. A class c_i is called **minority class**, if $\eta_i = p(c_i) < \frac{1}{k}$. Otherwise, we call c_i **majority class**.*

Hence, we distinguish as proposed in [15] three types of classification problems:

Definition 6. *Let γ_k be a classification problem with k classes and $\mathbf{1}(x)$ the indicator function (takes the value 1, if x is true, otherwise 0).*

1. *γ_k is called **balanced**, if η describes a uniform distribution.*
2. *γ_k is called **multi-majority**, if $\sum_{i=1}^{k} \mathbf{1}\left(\eta_i \geq \frac{1}{k}\right) \geq \frac{k}{2}$.*
3. *γ_k is called **multi-minority**, if $\sum_{i=1}^{k} \mathbf{1}\left(\eta_i < \frac{1}{k}\right) > \frac{k}{2}$.*

We estimate η by the empirical probability ζ from the existing dataset $\mathcal{D} = \{(x^{(n)}, c^{(n)})\}_{n=1}^{l}$, i.e.,

$$\eta_i = \zeta_i = \frac{1}{l} \sum_{n=1}^{l} \mathbf{1}\left(c^{(n)} = c_i\right).$$

Problem types 2 and 3 are known as *class imbalance problem* and are regarded as major obstacle for building precise classifiers [15]. The reason for that is that ML methods are designed to generalize the data in a suitable way [18]. Hence, they pay less attention to minority classes [18]. In addition to class imbalance, ML research also identifies the size of a dataset as further influencing factor [8,18]. However, a study in the process domain reveals, that the size of event logs nearly does not affect the quality of the classifier [12]. In general it is broad consensus in ML research that an approximately balanced dataset leads to better results. However, the imbalance must also be considered when selecting metrics for evaluating the classifiers, since some metrics are sensitive to the class distribution. From the application point of view the equal weighting of all errors is somehow unrealistic, since each error has its individual interpretation and consequences. Combining the problems of imbalance and equal weighting of misclassification errors, we identify three strongly related core issues:

Issue 1: Minority classes are not learned in an imbalanced dataset.
Issue 2: All misclassifications are treated and assessed equaly in an (approximately) balanced dataset.
Issue 3: Issues 1 and 2 jointly arise.

For an appropriate evaluation it is important to quantify class imbalance [15]. A frequently used metric is the so called *imbalanced ratio* (IR), which is defined as the number of examples of the largest class divided by the number of examples of the smallest class [8]. Hence, the IR is less suitable for multiclass problems, since only the largest and smallest classes are considered. In the following we use the *imbalance degree* (ID) that was proposed in [15] and considers all classes. The idea is to quantify the distance between the imbalanced class distribution ζ and a balanced class distribution e using statistical distance functions d_Δ (e.g., Kullback-Leibler divergence). The calculated distance is set in relation to the highest possible distance to e that a class distribution with the same number of classes and minority classes can have.

Definition 7. *The imbalanced degree is given by*

$$ID(\zeta) = \frac{d_\Delta(\zeta, e)}{d_\Delta(\iota_m, e)} + (m-1),$$

where m is the number of minory classes, d_Δ a statistical distance function, and ι_m is the class distribution with exactly m minority classes and the highest distance to e.

From Definition 7 follows that the ID takes a value within $[0, k)$. A high ID indicates a stronger imbalance than small values. The ID also includes explicitly the number of minority classes, since a high number of minority classes aggravates the problem. For calculating the ID, it is necessary to determine ι_m. According to [15], this class distribution consists of exactly m minority classes with probability zero, $k-m-1$ majority classes with probability $1/k$, and a majority class with the remaining probability of $1-(k-m-1)/k$.

3 Concept and Solution

3.1 General Idea

We address the issues described in Sect. 2.2 using *cost-sensitive learning* (CSL). While the conventional aim of training a ML model is to minimize the error (i.e., the number of incorrect predictions) of the model on a training dataset, CSL aims to minimize the cost of a model on a training dataset [8]. Therefore, each class is associated with a misclassification cost and the ML algorithms try to minimize the total misclassification cost [9]. The costs must be chosen in such a way, that the misclassification cost is always higher than correct classification [6]. The higher the costs of misclassification for a class, the more the training procedure must focus on examples of this class. We apply this concept to the three issues of the previous section as follows:

Issue 1: Misclassification errors of minority classes get assigned higher costs than those of majority classes.
Issue 2: Misclassification errors that produce high negative impact on the application get assigned high costs.
Issue 3: Costs stemming from Issue 1 and 2 must be combined.

Since, our concept is independent of a particular domain and imbalance is the prevalent problem, we focus on Issue 1 in the rest of the paper. However, the proposed concept is also applicable to the other issues and only requires an adaptation of the costs.

3.2 Calculating Class Weights

In the following, we assume that all classes are sequentially numbered from 1 to k. Let c_{ij} be the cost of predicting an example belonging to class j when in fact it belongs to class i. If $i = j$ the prediction is correct, otherwise the prediction is incorrect. This definition goes in hand with the assumption that the costs are only dependent on the classes and not on a concrete example [22]. The cost matrix $C = (c_{ij})_{1 \leq i,j \leq k}$ is defined as a $k \times k$ matrix with real values usually greater or equal to zero. In general, we can assume $c_{ii} = 0$, since a correct classification usually causes zero cost. The cost of the i-th class, i.e., the cost of misclassifying an example of class i regardless of the class predicted, is denoted by $C(i)$. Hence, a new example x should be classified to a class that minimizes the *conditional risk* (also called *expected cost*)

$$R(x|i) = \sum_{j=1}^{k} P(j|x)c_{ij}, \tag{1}$$

where $P(j|x)$ denotes the probability that the classifier classifies x as belonging to class j. In other words, R is the sum over all alternative possibilities for the true class of x [6]. Obviously, the cost matrix is crucial for the effectiveness of a CSL approach [8]. Wrong chosen costs hamper the learning procedure in several ways: *(i)* too low cost have no effect and the class boundaries are not adjusted, *(ii)* too high costs lead to overfitting, since they prevent a generalization of the model, and *(iii)* in the imbalanced case too high costs lead to an overcompensation and introduce a too strong bias towards the minority classes.

Hence, it would be optimal that domain experts create a cost matrix and derive the weight vector w. However, most of the time domain experts are not available, and the cost matrix must be created through heuristics. One option (called *Balanced Cost*) avoiding a cost matrix in the imbalanced case is to calculate class weights from the class distribution by $w_i = l/(k \cdot n(c_i))$, where $n(c_i)$ denotes the number of examples of the i-th class and l the number of examples of the dataset. Alternatively, we can use the IR for estimating a cost matrix. Therefore, we consider all pairs of classes seperately. That means, for two classes $i \neq j$, where i denotes the minority class and j the majority class, we calculate the IR between class i and class j and set c_{ij} to the calculated IR and c_{ji} to 1.

Given a cost matrix, we have two options for deriving the weight vector w:

Cost Sum: Here, all misclassfication costs related with a class are summed up [3,24], i.e., $w_i = C(i) = \sum_{j=1}^{k} c_{ij}$. However, the class weights are calculated isolated from other classes, why they are mostly not optimal. Hence, we try to derive the class weights simultanously for all classes from the cost matrix.

Optimized Costs: This procedure is based on the idea, that an optimal solution for w should also be an optimal solution in the binary case. Hence, we first study the binary case in more detail as it is done in [6,8,22,25]. An example x should be classified belonging to class 1, if and only if the expected costs (cf. Eq. 1) are lower than for classifying x as class 2, i.e.:

$$P(j = 1|x)c_{11} + P(j = 2|x)c_{12} \leq P(j = 1|x)c_{21} + P(j = 2|x)c_{22}.$$

Given $p = P(j = 1|x)$ this is equivalent to:

$$pc_{11} + (1 - p)c_{12} \leq pc_{21} + (1 - p)c_{22}.$$

If this inequality is in fact an equality, then the prediction is optimal. Assume that the optimal threshold is p^*. Then the following must hold:

$$p^*c_{11} + (1 - p^*)c_{12} = p^*c_{21} + (1 - p^*)c_{22}.$$

Since we can assume that c_{ii} is zero and $c_{ij} \neq 0$ for all $i \neq j$ the equation can be rearranged in the following way:

$$\frac{p^*}{1 - p^*} = \frac{c_{21}}{c_{12}} =: \lambda.$$

This means that the impact of the first class should be λ times of that of the second class. Hence, in the multiclass case this should hold for all $i, j \in \{1, ..., k\}$ with $i \neq j$:

$$\frac{w_i}{w_j} = \frac{c_{ji}}{c_{ij}} \Leftrightarrow w_i \cdot c_{ij} - w_j \cdot c_{ji} = 0.$$

Hence, we get a system of $\binom{k}{2}$ equations that is linear in w_i. This system has a non-trivial solution if and only if the rank of the corresponding coefficient matrix is smaller than k. In this case, we call the cost matrix *consistent* and calculate a solution with the QR decomposition. If the system does not satisfy this condition, it is not possible to derive an optimal weight vector.

3.3 Cost-Sensitive Transformation of Algorithms

Most ML algorithms are not cost-sensitive, i.e., they do not take the cost matrix into account [6,23]. Hence, it is necessary to use specialized CSL algorithms or transform existing algorithms into cost-sensitive ones. According to [8] we distinguish between *direct approaches* (DA) and *meta-learning approaches* (ME-A).

While direct approaches leverage the cost matrix directly in the training phase, the meta-learning approaches either modify the training data or the outputs of a trained classifier based on the cost matrix [8].

ME-A: Output Shift. In classifiers like neural networks that output a probability estimate $o = (o_1, ..., o_k)$, we can shift the output towards inexpensive classes [24]. For the probability estimate this equation holds: $\sum_{i=1}^{k} o_i = 1$ with $0 \leq o_i \leq 1$. Normally, the predicted class is determined by $\arg\max_i o_i$. For taking the cost matrix into account, we modify the probability estimate to

$$o_i^* = \delta \sum_{j=1}^{k} o_i c_{ij},$$

where δ is a normalization term such that $\sum_{i=1}^{k} o_i^* = 1$ and $0 \leq o_i^* \leq 1$. Hence, the cost matrix is only used in the test phase and does not modify the underlying data [24].

DA: Direct Approaches. If we want to use the cost matrix in the training phase directly, we must transform the classifiers individually into cost-sensitive ones. In case of neural networks as used for the process prediction, that means modifying the loss function used for training the neural network.

4 Related Work

Comparative studies reveal that deep learning approaches for next event prediction outperform classical ML techniques that use an explicit representation of the process model. Table 2 gives an overview of existing deep learning approaches for next activity prediction. Most of them apply Long-Short-Term Memory Neural Networks (LSTM) or Convolutional Neural Networks (CNN). Some other approaches are based on classical Deep Feedforward Networks (DFNN) or a specialized variant of a LSTM called Gated Recurrent Unit (GRU). These approaches take different event attributes into account (mostly the activity [ACT] and timestamp [Time]). Some of them also include derived event attributes, like roles or rules (i.e., process models). However, none of these approaches use CSL techniques or techniques for dealing with imbalance. Outside the process domain CSL gains attention since many years. In [6,25] the foundations of cost matrices are investigated and criteria for consistent cost matrices are derived. This concept is applied in [25] by an experimental study on 14 datasets of the *UCI ML repository* and 6 synthetic datasets (half of them imbalanced). In [8,9] the authors give an overview of the entire research of imbalanced learning. In [22] the concept of constant cost matrices is generalized to deal with cost intervals, that allows weighting particular examples differently. Also the usefulness of CSL in combination with different algorithms is thoroughly investigated: In [11] techniques of over- and undersampling are compared to CSL methods with regard to the class imbalance. Also they investigate, which classification algorithms are sensitive to class imbalance. In [24] methods (over- and

undersampling, threshold moving, hard and soft ensembles) for imbalanced classification with neural networks are compared on UCI datasets. Although over- and undersampling are an alternative to CSL, they bear some problems. In case of oversampling, pure duplicates have barely value, often the generated data is noisy, and training time increases. On the other hand, information gets lost by undersampling.

Table 2. Overview of existing deep learning approaches for next event prediction.

Approach	Network	Input	Prediction
Evermann [7]	LSTM	ACT	ACT
Mehdiyev [14]	DFNN	ACT, DP	ACT
Tax [19]	LSTM	ACT, Time	ACT, Role, Suffix, Time
Al-Jebrni [2]	CNN	ACT	ACT
Schönig [17]	LSTM	ACT, ATTR	ACT, Resource
Camargo [4]	LSTM	ACT, Role, Time	ACT, Role, Time, Suffix
Lin [13]	LSTM	ACT, DP	ACT, DP, Suffix
Hinkka [10]	GRU	ACT, DP	ACT
Theis [21]	DFNN	ACT, Rules, Time, DP	ACT
Pasquadibisceglie [16]	CNN	ACT, Time	ACT
Mauro [5]	CNN	ACT, Time	ACT

5 Implementation and Evaluation

5.1 Dataset Description and Experimental Setup

We evaluate our concept by combining two state-of-the-art approaches for next activity prediction with the procedures described in Sect. 3. We select approaches [4] and [16], since they represent the most frequently used network architectures (LSTM and CNN respectively), and achieve good results in comparative studies. We build our implementation[1] upon those of the considered approaches using Python 3.7 and run the neural networks with Tensorflow, which we use at version 2.1.0 with GPU support via CUDA for expression evaluation. We perform our experiments using 4 real-life event logs from different domains with diverse characteristics (cf. Table 3) extracted from the *4TU Center for Research Data*[2]. However, the event logs Sepsis and Traffic do not record the performing resource,

[1] The source code can be accessed at https://github.com/mkaep/.
[2] https://data.4tu.nl/.

so they cannot be evaluated with approach [4]. The splitting of the event logs into training and testdata is done along the time dimension, by ordering the traces ascending by its first event timestamp and then selecting the first 70% of the traces for training. For evaluation, we use the following common metrics for imbalanced datasets that can be derived using the confusion matrix[3]: *(i) Recall* (also called sensitivity) defined as $R = TP/(TP + FN)$, *(ii) Precision* $P = TP/(TP + FP)$, and *(iii) F-Measure* defined as harmonic mean $F = 2RP/(R + P)$ of precision and recall. As recommended, we first calculate the metrics per class and afterwards, we calculate the arithmetic mean of the per class scores [8]. Hence, we weight the classes equally, independently of their probability. Additional, we use the *geometric mean* defined as $G = \sqrt{R_{avg}S_{avg}}$ the squared root of the averaged recall and specificity. The specificity can be calculated by $S = TN/(TN+FP)$. Furthermore, we quantify for each approach the imbalance of the dataset, by determining the number of minority classes (m), IR, and ID as recommended in [15]. The cost matrices are derived with the heuristic method described in Sect. 3.2. The experiments were run on a system equipped with a Windows 10 operating system, an Intel Core i9-9900K CPU3.60 GHz, 64 GB RAM, and a NVIDIA Quadro RTX 4000 having 6 GB of memory.

Table 3. Statistic of the used event logs.

Event log	BPIC12	Helpdesk	Sepsis	Traffic
Number of cases	9559	4580	1050	150370
Number of activities	36	14	16	11
Maximal case length	163	15	185	20
Minimal case length	3	2	3	2
Average case length	14.74	4.66	14.49	3.73

5.2 Overall Results and Further Analysis

We discuss now our results in detail. All measures are shown in Table 4. Note, that measuring the metrics does not primarily evaluate the concept of CSL itself rather than the quality of the cost matrices. Considering m and the IR, reveals that all event logs are multi-minority problems (cf. Definition 6) with strong imbalance. Normalizing the ID shows differences between the event logs and some minor ones between the applied approaches (caused by the different minimum length of event prefixes). Unfortunately, in all cases the cost matrix is inconsistent, so it is not possible to derive an optimal weight vector. Thus results for this method cannot be proclaimed. In case of the Sepsis log the *Balanced Cost* method lowers the performance. On all other logs the performance of

[3] i.e., it can be calculated from true positives (TP), true negatives (TN), false positives (FP), and true positive (TP).

Table 4. Overall results (best values are bold faced).

		BPIC12		Helpdesk		Sepsis		Traffic	
		[16]	[4]	[16]	[4]	[16]	[4]	[16]	[4]
Recall									
	Conventional	0.67	0.67	0.17	0.29	0.47	–	0.70	–
OS	*Balanced Cost*	0.65	0.66	0.21	0.28	0.36	–	0.63	–
	Cost Sum	0.75	0.73	**0.29**	**0.34**	**0.55**	–	**0.76**	–
DA	*Balanced Cost*	0.70	0.67	0.22	0.24	0.38	–	0.71	–
	Cost Sum	**0.76**	**0.75**	0.28	0.32	0.53	–	0.74	–
Precision									
	Conventional	0.72	0.76	0.13	0.26	0.48	–	0.74	–
OS	*Balanced Cost*	0.71	0.75	0.17	0.27	0.24	–	0.65	–
	Cost Sum	**0.78**	0.78	**0.21**	**0.31**	0.53	–	**0.79**	–
DA	*Balanced Cost*	0.68	0.75	0.15	0.25	0.29	–	0.70	–
	Cost Sum	**0.78**	**0.79**	**0.21**	0.30	**0.54**	–	0.78	–
F-Measure									
	Conventional	0.67	0.68	0.14	0.27	0.47	–	0.70	–
OS	*Balanced Cost*	0.65	0.61	0.09	0.28	0.12	–	0.51	–
	Cost Sum	0.74	**0.73**	**0.21**	**0.32**	0.52	–	**0.76**	–
DA	*Balanced Cost*	0.68	0.64	0.15	0.26	0.21	–	0.69	–
	Cost Sum	**0.75**	**0.73**	0.20	**0.32**	**0.55**	–	0.74	–
Geometric Mean									
	Conventional	0.81	0.82	0.40	0.53	0.67	–	0.83	–
OS	*Balanced Cost*	0.80	0.81	0.47	0.55	0.46	–	0.69	–
	Cost Sum	0.85	0.85	0.48	**0.59**	0.73	–	0.87	–
DA	*Balanced Cost*	0.83	0.81	0.48	0.51	0.57	–	0.85	–
	Cost Sum	**0.87**	**0.86**	**0.50**	0.58	**0.74**	–	**0.89**	
#Classes/m		34/18	37/21	13/9	14/9	16/10	–	10/6	–
Imb. Ratio (IR)		4011	2326	4702	3717	556	–	145	–
Imb. Degree (ID)		17.48	20.46	8.85	8.80	9.65	–	5.75	–
Normalized ID		0.51	0.55	0.68	0.63	0.60	–	0.57	–

the Balanced Cost method is for the output shift (OS) and the direct approach (DA) nearly identical to the performance of conventionally trained classifiers. Only some minor deviations can be observed. The *Cost Sum* method, however, achieves a respectable performance gain over all considered metrics between 3% and 10% and can be beneficially applied for both the output shift approach and the direct approach. Hence, this approach slightly outperforms the conventional ones. Furthermore, we observe usefulness of CSL for both predictive business

process monitoring approaches. In general, we observe, that a less extreme imbalance results in better classification results. Hence, we can confirm the results of studies (cf. Sect. 4) in other domains also for the process domain. Since, the cost matrix is not consistent CSL has only a moderate but notable impact. Hence, we draw the following conclusions: The achieved performance seems promising for further research which should focus on the mining of consistent cost matrices. Furthermore, it should be investigated whether methods of numeric mathematic for calculating approximate solutions of overdetermined systems can be used to calculate a satisfactory weight vector.

6 Conclusion and Future Work

In this paper, we propose a concept for leveraging CSL for the prediction of categorical event attributes to cope the problem of an imbalanced class distribution. Our experiments reveal that CSL can be beneficially applied for next activity prediction. In addition to the above mentioned issues for future work, we plan to use CSL for binary problems like outcome oriented predictive process monitoring, since in the two class case the equation system for calculating an optimal weight vector is more often solvable. Also, it should be investigated how the performance differs from those of alternative options like over- and undersampling and, whether a combination of both, can achieve better results.

References

1. van der Aalst, W.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
2. Al-Jebrni, A.H., Cai, H., Jiang, L.: Predicting the next process event using convolutional neural networks. In: IEEE International Conference on PIC, pp. 332–338 (2018)
3. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. Taylor & Francis (1984)
4. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 286–302. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_19
5. Di Mauro, N., Appice, A., Basile, T.M.A.: Activity prediction of business process instances with inception CNN models. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019. LNCS (LNAI), vol. 11946, pp. 348–361. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35166-3_25
6. Elkan, C.: The foundations of cost-sensitive learning. In: Proceedings of IJCAI 2001, pp. 973–978. Morgan Kaufmann Publishers Inc., San Francisco (2001)
7. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**, 129–140 (2017)
8. Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F.: Learning from Imbalanced Data Sets. Springer, Heidelberg (2018)
9. He, H., Ma, Y.: Imbalanced Learning: Foundations, Algorithms, and Applications, 1st edn. Wiley-IEEE Press (2013)

10. Hinkka, M., Lehto, T., Heljanko, K.: Exploiting event log event attributes in RNN based prediction. In: Welzer, T., et al. (eds.) ADBIS 2019. CCIS, vol. 1064, pp. 405–416. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30278-8_40
11. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. Intell. Data Anal. **6**(5), 429–449 (2002)
12. Käppel, M.: Evaluating predictive business process monitoring approaches on small event logs. arXiv (2021)
13. Lin, L., Wen, L., Wang, J.: A deep predictive model for multi-attribute event sequence. In: Proceedings of the International Conference on Data Mining 2019, pp. 118–126 (2019)
14. Mehdiyev, N., Evermann, J., Fettke, P.: A multi-stage deep learning approach for business process event prediction. In: IEEE 19th CBI, pp. 119–128 (2017)
15. Ortigosa-Hernández, J., Inza, I., Lozano, J.A.: Measuring the class-imbalance extent of multi-class problems. Pattern Recogn. Lett. **98**, 32–38 (2017)
16. Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Using convolutional neural networks for predictive process analytics. In: Proceedings of ICPM 2019 (2019)
17. Schönig, S., Jasinski, R., Ackermann, L., Jablonski, S.: Deep learning process prediction with discrete and continuous data features. In: Proceedings of ENASE 2018 (2018)
18. Sun, Y., Kamel, M.S., Wong, A.K., Wang, Y.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn. **40**(12), 3358–3378 (2007)
19. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30
20. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. TKDD **13**(2) (2019)
21. Theis, J., Darabi, H.: Decay replay mining to predict next process events. IEEE Access **7**, 119787–119803 (2019). https://doi.org/10.1109/ACCESS.2019.2937085
22. Wu, J., Wan, L., Xu, Z.: Algorithms to discover complete frequent episodes in sequences. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 7104, pp. 267–278. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28320-8_23
23. Zhang, X., Hu, B.: A new strategy of cost-free learning in the class imbalance problem. IEEE Trans. Knowl. Data Eng. **26**(12), 2872–2885 (2014)
24. Zhou, Z.-H., Liu, X.-Y.: Training cost-sensitive NN with methods addressing the class imbalance problem. IEEE Trans. Knowl. Data Eng. 63–77 (2006)
25. Zhou, Z.H., Liu, X.Y.: On multi-class cost-sensitive learning. In: Proceedings of the 21st National Conference on AI, pp. 567–572. AAAI Press (2006)