Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 161 (2019) 1080–1088

Procedia Computer Science

www.elsevier.com/locate/procedia

The Fifth Information Systems International Conference 2019

Predictive Business Process Monitoring – Remaining Time Prediction using Deep Neural Network with Entity Embedding

Nur Ahmad Wahid[a], Taufik Nur Adi[b], Hyerim Bae[b,*], Yulim Choi[b]

[a]*Department of Big Data, Pusan National University, Busan 609-735, South Korea*
[b]*Department of Industrial Engineering,Pusan National University, Busan 609-735, South Korea*

Abstract

Most process mining study focuses on analysis of past data. This differs from predictive process monitoring, which, as a part of operational support, has as one of its focuses on the prediction of a running case [1]. Although there are several measures of interest that can be provided, in the present study, we focused on the remaining time of a running case. Results produced by Deep Neural Network (DNN) [2], despite its acknowledged power for various problems, typically are no better than those of other supervised algorithms with problems involving categorical variables in tabular data [3]. Because the dataset extracted from event logs that contain categorical variables can be constructed and categorized in tabular form, it is unwise to use only ordinary DNN. In this study, we showed that we can increase the accuracy of DNN on tabular data that contains categorical variables by using a technique known as Entity Embedding. To show the robustness of the method, we conducted experiments with two types of dataset, synthesis data and real-world data, and also compared its performance with other supervised learning algorithms for regression problems. The experimental results showed that it is true that the proposed method can increase the accuracy of DNN predictions on remaining time prediction problem involving categorical variables and beats all baseline methods used as comparison.

Keywords: predictive process monitoring; deep neural network; entity embedding

* Corresponding author. Tel.: +82-051-510-27331.
E-mail address: hrbae@pusan.ac.kr

10.1016/j.procs.2019.11.219

1. Introduction

Predictive business process monitoring is a study framework that aims to provide predictions of some measures of interest of ongoing cases based on an event log of past processes. Examples of measures of interest are the activity that will occur next in a running process and the remaining time required by a process for its completion. In this paper, we focus on providing prediction of remaining time for a running case. There are many cases in which it would be very useful to have good remaining-time predictions. By acquiring knowledge in advance about a process that is still running, the manager or other person in charge can take action to avoid undesirable situations; as such, remaining time is indeed very useful for effective management of business processes. What is more, the client or customer also can benefit. One example is a customer's call to his insurance company to ask about when he can claim his insurance. In this case, he can be given an estimate of the remaining time required by the claim process [4]. Another example can be found in the port logistics process which has activities such as loading and discharging containers, both by the yard truck and quay crane. By providing accurate information about how much longer it will take for an ongoing process to complete, or how long it takes for a yard truck to load and discharge the container, all of that can help the port authorities in decision making and ultimately improve the efficiency and effectiveness of the logistics management port. The remaining-time prediction problem can be categorized as supervised learning, wherein we have a dataset, X is representing features of training data, Y indicating label or output, (x_i, y_i) being a training pair with $i = 1 \dots N$, y_{pred} being the predicted label value, and F being a risk function such as Root Mean Squared Error (RMSE). Our goal would be to optimize Eq. 1.

$$\text{minimize} \frac{1}{N} \sum_{i=1}^{N} F(y_{pred_i} - y_i) \tag{1}$$

There are already some researches on this topic. Van Dongen et al. used a non-parametric regression technique to predict the remaining time [5]. Aalst et al. used a process mining technique to predict time-related outcomes [4]. M. Ceci et al. proposed a similar approach but their method can also predict the next activity [6]. Leontjeva et al. utilized the Hidden Markov Model to extract complex features for more accurate predictions [7]. Polato et al. used various machine-learning models such as Naive Bayes Classifier and Support Vector Regression to predict various measures of interest [8]. Senderovich et al. conducted heavy feature engineering to extract inter-case features from a dataset [9]. In the present study, we employed Deep Neural Network (DNN) with a basic architecture (namely deep feed forward with a fully connected layer) in combination with the Entity Embedding technique to significantly improve the accuracy of remaining-time prediction.

2. Preliminaries

2.1. Event log

Process mining requires event logs as input. In practice, event logs can be very different. However, all event logs have a common characteristic, in that they show occurrences of events at specific moments in time, where each event refers to a particular process and an instance thereof, i.e., a case. Table 1 is an event log fragment that illustrates the typical information in an event log. In process mining, any event can be related to both a case and an activity, and the events within a case are orders. Therefore, the "case id" and "activity" columns in Table 1 represent the bare minimum for process mining. As shown in Table 1, case 1 has five associated events. Each line corresponds to an event of a case. The first event of Case 1 is the execution of an activity register request. The unique id in the "Event id" column is used for the identification of the event, e.g., to distinguish it from event 1283, which also corresponds to the execution of an activity register request.

Table 1. A fragment of event logs.

Case id	Event id	Timestamp	Activity
1	1223	30-12-2012:11.02	register request
	1224	31-12-2012:10.06	examine thoroughly
	1225	05-01-2013:15.12	check ticket
	1226	06-01-2013:11.18	decide
	1227	07-01-2013:14.24	reject request
2	1283	30-12-2012:11.32	register request

2.2. Neural network

Neural network, also known as artificial neural network, is a computational model inspired by the structure and functions of a biological neural network. The basic processing elements in a biological neural network are based on the neurons in the brain and the synapses between them. The nodes (represented neurons) and the weighted links between them are the basic elements in the standard neural network. Processing occurs through the spread of activation from nodes along weighted links. Nodes in the network receive activations from other nodes that are connected to them. The level of activation is determined by a function that calculates the multiplication of all sending nodes and the weight on the link between the sending and the receiving node. The node sums the activation from all of the sending nodes and then sends activations to subsequent nodes based on this sum. The output for a node can take several forms. First, the output can be binary, whereby a node sends an activation when the threshold is exceeded or, otherwise, does not. Second, the output activation can be a linear function of the input. Third, it can be an S-shaped or sigmoidal function of the inputs.

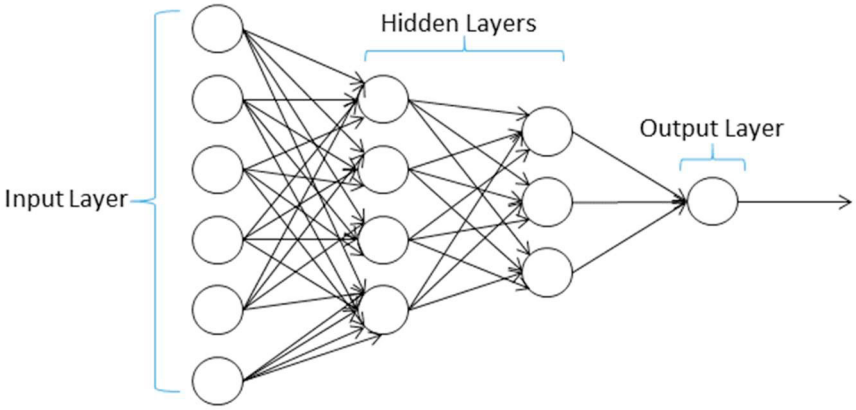

Fig. 1. Neural Network [10].

Fig. 1 shows the basic architecture of a neural network. The leftmost layer in the network is called the input layer, and the nodes within the layer are called input nodes. The middle layers are called hidden layers, which are layers in the processing stream between input and output. These kinds of hidden layers can learn more complex representations that are combinations of lower-level feature. The rightmost layer, called the output layer, contains the output nodes.

2.3. One hot encoding

One known method in machine learning for treating categorical variables is to encode them into integer labels. However, that is not sufficient, because our machine-learning model will assume that a higher label number has a higher influence than a lower label number. Therefore, we need an encoding method that assigns equal weight to

categorical variables. Thus, some labels do not have higher weight than other labels. This encoding is known as One Hot Encoding [10]. Fig. 2 shows the illustration, whereby variables will be encoded in binary form. These result variables are also called dummy variables. One feature will be broken down into as many cardinals as the feature, with only one of them having a value of one, the others having a value of zero.

COLOUR
Red
Green
Blue

Red	Green	Blue
1	0	0
0	1	0
0	0	1

Fig. 2. One Hot Encoding.

2.4. Embedding technique

The One Hot Encoding technique has some disadvantages, such that if a feature has variables with a high cardinality level, the number of binary columns produced will be very large and, as a result, the dataset becomes very sparse or has very high dimensions. Not only that, values that are similar to each other are not placed close together in space, the result being that the intrinsic properties of the feature are eliminated. One approach that can overcome these problems is to use a technique known as Entity Embedding. This technique is also similar to one, known as "word embedding," that is used in Natural Language Processing. Word embedding works by mapping words and phrases to vectors in multi-dimensional space so that similar words will be placed close to each other in that space. This embedding space can be obtained by learning methods. There has been a lot of research conducted to study how to acquire the embedding. One fast way [12] is to use the context that words have by maximizing where W is a vector that represents the word w and the word w_c is the neighbor word in the context window that has been defined. While $p(w_c|w)$ is the possibility to have the word w_c in the context window of the word w (See Eq. 2).

$$p(w_c \mid w) = \frac{\exp(W \cdot W_c)}{\sum_i \exp(W \cdot W_c)} \tag{2}$$

Value of Categorical Feature	Embedding
value_0	[1.0, 0.0, 3.0]
value_1	[2.0, 0.0, 1.0]
value_2	[1.0, 3.0, 0.0]
value_3	[2.0, 1.0, 0.0]
value_4	[1.0, 0.0, 4.0]

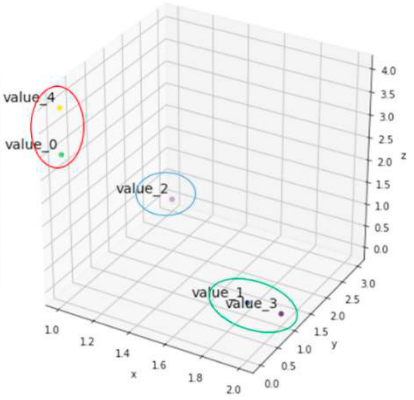

Fig. 3. Categorical variables into embedding space.

By using this encoding method, values that have similar intrinsic properties will be placed close together in space (Fig. 3). Not only that, memory usage will be reduced due to the fact that the dimensions of features are smaller than those of One Hot Encoding, which will indirectly increase the speed of the DNN.

3. Proposed method and implementation

3.1. Problem and data preprocessing

Remaining-Time Prediction is a supervised learning problem. Therefore, we need to first define the inputs (or features) and the output. The inputs are all of the attributes that an event has, such as categorical variables and time-related features (continuous values) such as elapsed time. We also need to embed a history of the sequence of events from the beginning of the case to the event currently examined. The output, meanwhile, is the remaining time needed for the case to complete, which is the measure of interest we focused on.

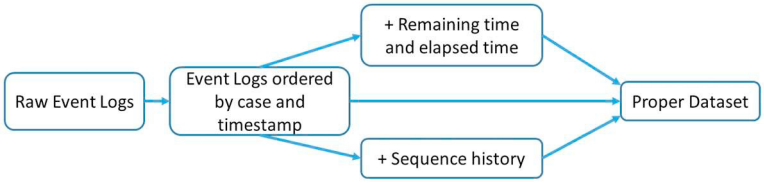

Fig. 4. Steps to pre-process the event logs.

Like analytical problems in general, we process the raw data first so that it becomes a proper dataset for further analysis. The process starts by sorting the raw event log by case and timestamp, and then proceeds to embed the time-related features such as elapsed time and remaining time. Elapsed time is the time spent from the start of the case to the end, while remaining time is the time left from the current state to the end of the case. These two-time features can be calculated using only the timestamp that the event log has. The last step is to insert the sequence history of events from the initial state to the current state (Fig. 4).

3.2. Deep neural network + entity embedding

We implemented a fully connected neural network [10] as the basic architecture of the model. Every continuous variable is directly connected to the first hidden layer, while for each of the categorical variables, an embedding layer is created (Fig. 5). In our DNN, we implemented five hidden layers. In each hidden layer, we applied the ReLU activation function [13]. ReLU is the most popular activation function nowadays, owing to its efficiency and ease of use in the computing process. For learning algorithm, we adopted Adam optimizer [14].

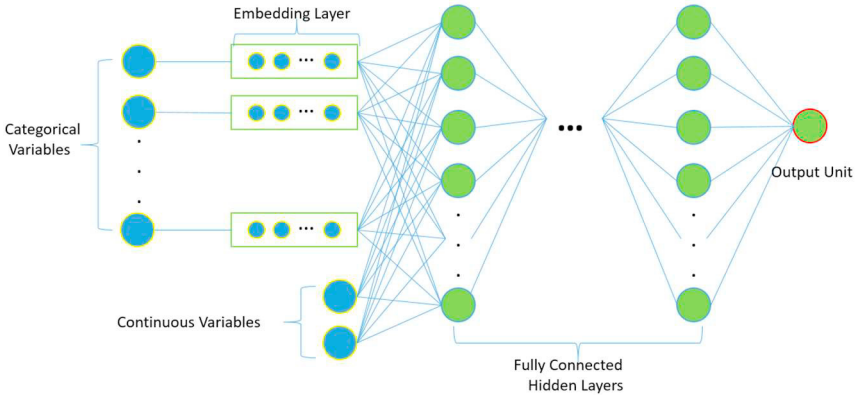

Fig. 5. Deep Neural Network with Entity Embedding.

As for the number of neurons in each hidden layer, we set it high at the beginning of the layer and decrease it at the end of the layer, according to the manner in which neural networks generally are implemented. In the training process, to help avoid overfitting conditions, we applied the validation test technique with Early Stopping [15] so that the training process would stop when our model showed indications of overfitting conditions. We also saved weights at the optimal times, namely when validation loss occurred at the lowest point. This is shown in Fig. 6 with the red rectangle. All of our implementations were done in Python version 3.6 using the Keras [16] library with Tensorflow [17] at the backend.

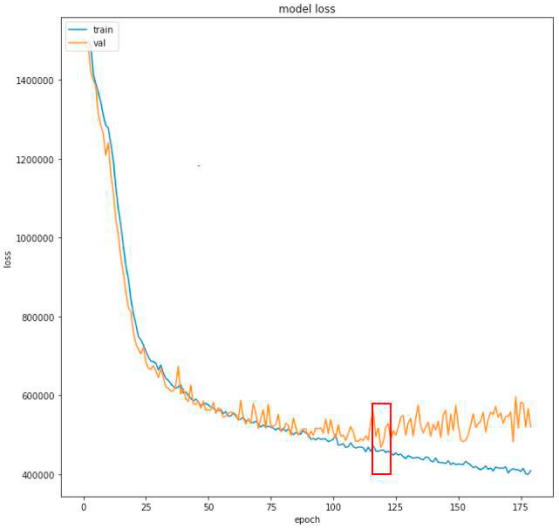

Fig. 6. Training Process with Early Stopping.

4. Experiments and findings

In this section, we will show the experiments we conducted to demonstrate the accuracy of our proposed model's performance. To demonstrate the robustness of the model, we conduct experiments with two types of datasets, a real-world dataset and a synthesis dataset. For each dataset, we divide it randomly 80:20 into a training dataset and a test dataset. We measured the performance of the proposed method also by comparing it with other regression methods using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). RMSE is the default error metric standard used by various models. However, RMSE is rather sensitive to outliers [18], due to the fact that higher difference is penalized more than with MAE. The methods used for comparison were ordinary Fully Connected DNN, DNN with One Hot Encoding, and regression method packages supplied by the Scikit Learn library [19], namely Linear Regression, Ridge, Lasso, and SVR. Linear regression is the basic machine learning method for regression problems. Ridge and Lasso are extensions of Linear Regression and are basically regularized linear regression but they both have regularization technique that are different from each other, while SVR [20] is a Support Vector Machine method for regression problem. Please note here that the DNN architecture and its hyper-parameters are exactly the same as those of the DNN we used with Entity Embedding. All the experiments were conducted on a 4 core Intel® Core™ i7-4790K CPU @ 4.00GHz each core with 16GB RAM, running on Windows Server 2016 Datacenter.

4.1. Real world dataset

For the real-world dataset, we used the event log of a port logistics process containing six kinds of activities such as loading and discharging containers, both by the yard truck and quay crane. and only three events for each case. The data had five categorical features besides the activity itself and two timestamps (the start and the end of times of each event). After preprocessing the data, we obtained one continuous feature, the elapsed time, and a measure of interest,

the remaining time of a running case. We also produced six additional features related to sequence history and treated them as categorical features. We show the results in Table 2. As can be seen, DNN with Entity Embedding was best in terms of accuracy; it was slightly better than DNN with One Hot Encoding. All of the regression methods except DNN and SVR showed similar results. SVR was worst according to this experiment.

Table 2. Experimental results for Port Logistics Process Event Log and Synthesis Event Log in RMSE (minutes) and MAE (minutes).

	Port Logistics Process Event Log		Synthesis Event Log	
	RMSE	MAE	RMSE	MAE
DNN	427.22	233.62	33508.63	21660.89
DNN + One Hot Encoding	380.09	198.96	24300.96	15418.45
DNN + Entity Embedding	364.95	190.32	14015.11	5913.58
Linear Regression	399.73	249.92	25133.18	18278.28
Ridge	399.73	249.92	25131.96	18278.24
Lasso	400.53	253.39	25133.40	18279.63
SVR	593.51	469.44	33406.49	23000.13

4.2. Synthesis dataset

This dataset is an event log we obtained from the work of A. Senderovich et al. It consists of only four columns, case id, timestamp, event or activity, and resource. Besides the number of columns, the contrasting difference with the port logistics process data is that a case can consist of ~30 events. Because there are many events that can be contained by one case, the preprocessing stage makes the data more dimensional than the port logistics process data. In comparison, the real-world data had 13 features after preprocessing. It was very different from synthesis data, which had 352 features on the same stage. Furthermore, the more categorical features, the more the accuracy of the DNN + Entity Embedding technique increased, as we show in Table 2. DNN + Entity Embedding was much more effective than the other methods for this dataset. This was due to the fact that the dataset had more features after the preprocessing stage compared with the real-world dataset. Just as with the real-world dataset, DNN showed increased accuracy after the categorical variables were encoded with One Hot Encoding. Once again, SVR showed the worst performance.

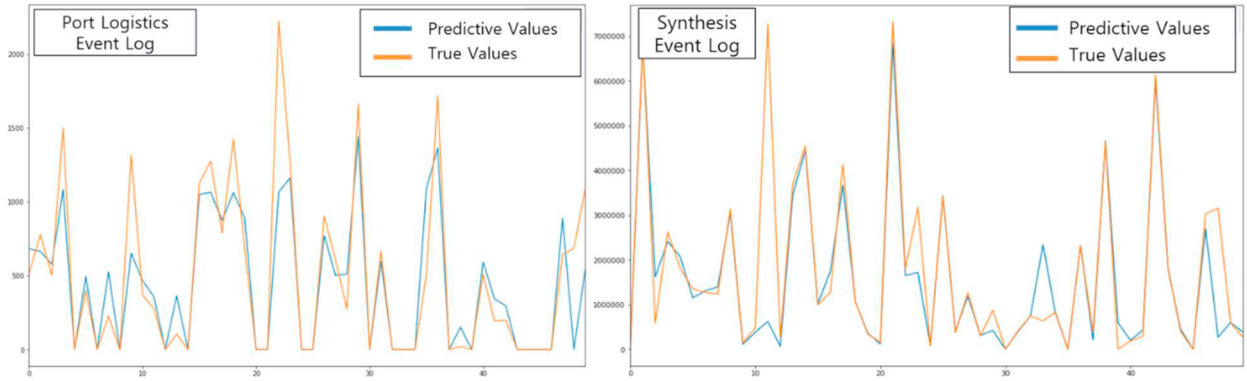

Fig. 7. Overview Comparison between True Values and Predictive Values on The First 50 Test Data using DNN + Entity Embedding.

In Fig. 7, we provide a comparison of the results of predictions with the real value of the first 50 data tests using DNN + Entity Embedding in minutes (unit of measurement time).

5. Conclusion and future works

In this study, we tested a proposed predictive model using the DNN + Entity Embedding technique for prediction of a measure of interest in predictive business process monitoring, namely the remaining time of an ongoing case. The proposed method proved to be able to increase the accuracy of DNN significantly and to predict accurately, as shown in experiments. For datasets having low dimensions, DNN + Entity Embedding was only slightly better than the other methods, whereas for datasets having high dimensions, it was much better, as indicated by the magnitude of the difference in error metrics. Based on the result, the proposed method is very well applied to the problem of remaining time prediction which involves many categorical variables because of the nature of Entity Embedding technique that places categorical values that have intrinsic properties that are similar to each other in multi-dimensional space. Considering that machine learning, especially DNN, has many hyper-parameters such as the number of layers and the number of neurons in each layer, in the future, we plan to implement an optimization method that optimizes the hyper-parameter configuration. We will also perform another experiment, this one entailing extraction of the embedding layer and its use as features in other models. As a future study, we plan to not only predict the remaining time of ongoing process, but also how to predict what is its next activities (one of the prediction problem in Predictive Business Process Monitoring). It is possible to extend the method to deal with that problem in a multi-task prediction manner, such as the work of N. Tax et al. [21], but their method does not consider data attributes of event logs.

Acknowledgements

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Consilience Creative program (IITP-2019-2016-0-00318) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation).

References

[1] Aalst, W. Van Der. (2016) *Process Mining: Data Science in Action*, Second Edition, Eindhoven, Springer.
[2] LeCun, Y., Y. Bengio, and G. Hinton. (2015) "Deep Learning." *Nature* **521 (7553)**: 436-444.
[3] C. Guo, and F. Berkhahn. (2016) "Entity Embeddings of Categorical Variables." *CoRR* **1604 (06737)**.
[4] Aalst, W. Van Der, M. H. Schonenberg, and Song M. Time. (2011) "Time Prediction Based on Process Mining." *Information System*.
[5] Dongen, B.F. van, R.A. Crooy, and W.M.P. van der Aalst. (2008) "Cycletime Prediction: When Will This Case Finally Be Finished?" *Cooperative Information Systems*.
[6] M. Ceci, P. F. Lanotte, F. Fumarola, D. P. Cavallo, and D. Malerba (2014) "Completion Time and Next Activity Prediction of Processes Using Sequential Pattern Mining." *Discovery Science Conference*.
[7] Leontjeva, A., R. Conforti, C. D. Francescomarino, M. Dumas, and F. M. Maggi. (2015) "Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes." *Business Process Management*.
[8] Polato, M., A. Sperduti, A. Burattin, and M. de Leoni. (2016) "Time and Activity Sequence Prediction of Business Process Instances." *arXiv preprint*.
[9] A. Senderovich, C. Di Francescomarino, C. Ghidini, K. Jorbina, and F. M. Maggi. (2017) "Intra and Inter-Case Features in Predictive Process Monitoring: A Tale of Two Dimensions", in *Business Process Management - 15th International Conference BPM.*
[10] Goodfellow, I., Y. Bengio, and A. Courville. (2016) "Deep Learning." *MIT Press*.
[11] Beck, J.E., and B. P. Woolf. (2000) "High-level Student Modeling with Machine Learning." *Intelligent tutoring systems*, *Springer*.
[12] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean (2013) "Distributed Representations of Words and Phrases and Their Compositionality", in *Advances in Neural Information Processing Systems*. pp. 3111–3119.
[13] Glorot, X., A. Bordes, and Y. Bengio. (2011) "Deep Sparse Rectifier Neural Networks." in *AISTATS*.
[14] Kingma, Diederik P., and Jimmy Lei Ba. (2015) "Adam: A method for stochastic optimization" in *International Conference for Learning Representations*.
[15] Yao, Yuan, Lorenzo Rosasco, and Andrea Caponnetto. (2007) "On Early Stopping in Gradient Descent Learning." *Constructive Approximation*.
[16] Chollet, F. (2015) "Keras." Available from: https://github.com/fchollet/keras.
[17] Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin, et al. (2016) "TensorFlow: Large-scale Machine Learning on Heterogeneous Distributed Systems." *arXiv Preprint* **1603(04467)**.
[18] Chai, T., and R.R. Draxler. (2014) "Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? – Arguments Against Avoiding RMSE in the Literature." *Geoscientific Model Development*.

[19] Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, and P. Prettenhofer, et al. (2011) "Scikit-learn: Machine Learning in Python." *The Journal of Machine Learning Research* **12**: 2825–2830.
[20] Drucker, Harris, Chris J.C. Burges, Linda Kaufman, Alex Smola, and ladimir Vapoik. (1997) "Support Vector Regression Machines."
[21] Tax, N., I. Verenich, M. L. Rosa, and M. Dumas. (2017) "Predictive Business Process Monitoring with LSTM Neural Networks." *CAiSE*.