Applied Predictive Process Monitoring and Hyper Parameter Optimization in Camunda

Nico Bartmann, Stefan Hill(✉), Carl Corea, Christoph Drodt, and Patrick Delfmann

Institute for Information System Research, University of Koblenz-Landau, Koblenz, Germany
{nbartmann,shill,ccorea,drodt,delfmann}@uni-koblenz.de

Abstract. With the growing amount of data gathered from business processes in recent years, predictive process monitoring (PPM) established as a method to provide valuable insights and make resilient forecasts. However, sophisticated machine learning algorithms and statistical techniques are always equipped with various hyper parameters, which aggravates finding the best configuration for laypeople. Tools like Nirdizati Research (http://research.nirdizati.org/) or apromore (https://apromore.org/) aim to assist in these tasks. Nonetheless these approaches are isolated solutions, which do not integrate into existing productive environments. In this work, a plugin for the widely used workflow and decision automation tool Camunda (https://camunda.com/) is presented which allows creating classifier for the most common operations in PPM. Furthermore, the framework includes a hyper parameter optimization (HPO) and is extensible in prediction types, methods and optimization algorithms.

Keywords: Predictive process monitoring · Hyper parameter optimization · Camunda

1 Introduction

In the last two decades, the digitalization of business processes has equipped companies with substantial new means to investigate the internal company processes. With business processes being executed by workflow management systems that continuously log event data, the availability of big data allows to generate valuable insights into company activities. Here, a wealth of recent research has focused on exploiting such workflow log data for predictive process monitoring, in order to leverage insights to create competitive advantages by means of intelligent predictions (c.f. [5] for an overview). While such results are clearly beneficial for companies, there are unfortunately no user friendly solutions for PPM that integrate seamlessly into existing workflow management systems and guide end-users through the selection of suitable prediction methods. Even though there

S. Nurcan and A. Korthaus (Eds.): CAiSE Forum 2021, LNBIP 424, pp. 129–136, 2021.
https://doi.org/10.1007/978-3-030-79108-7_15

are refined solutions for PPM like for example Nirdizati Research or several ProM-Plugins[1], these are always implemented in an isolated environment. This however impedes a practical usage, as running a workflow engine and analysing tools in parallel can lead to data storage problems such as redundancy and requires extensive employee training for the external tools. Here, methods are needed that integrate PPM with existing workflow management systems in a unified manner, in order to lower the obstacles of adoption.

In this work, we therefore introduce a Camunda plugin[2] which enables an effortless use of machine learning techniques for PPM directly in the running workflow management system. Camunda is an open source workflow management and automation platform which is widely used by a variety of small- to large-scale companies such as Atlassian, Generali and Deutsche Telekom[3]. The presented plugin adds a clean user interface for predictions on single process instances and a detailed configuration panel for administrative tasks. Classifiers may be trained on the basis of the internal Camunda log or with external logs in the common Extensible Event Strean (XES) format. The plugin ships with three available prediction types of next activity, time and risk prediction that can directly be used, however, arbitrary prediction types like forecasting process variables or cost can be added afterwards. Also, the Classifier interface is left generic to allow for an extension with arbitrary prediction algorithms. Out of the box, N-Grams, LSTM neural networks, regression are available inter alia.

While the flexible design allows for a high degree of customization, the wealth of prediction types and prediction algorithms may require extensive background knowledge by end-users to select an optimal prediction method and therefore might hamper the feasibility in practice due to the skill sets of the involved (non-technical) users. Therefore, the core feature of this plugin also includes HPO for combined algorithm search and hyper parameter setting. Classifiers are ranked by an evaluation metric and may be created on the fly. In this way, the optimal classifier type and parameter settings for the company can be determined by the plugin. This is especially useful for the training of predictive models for individual company logs, as the sheer amount of algorithms and hyper parameter settings largely depends on the process definition and the underlying log, which makes it impossible to provide generally suitable parameter settings.

This article will continue with a background section on PPM and optimization. Then, the approach architecture will be introduced in Sect. 3, including data management, different classifier architectures and HPO techniques. Afterwards, the implemented plugin will be demonstrated by showing typical usage examples. To evaluate the feasibility of our approach, results of conducted runtime experiments on real-life datasets will be presented in Sect. 4. Last, the contributions and limitations of this work are concluded in Sect. 5.

[1] http://promtools.org/.
[2] https://gitlab.uni-koblenz.de/fg-bks/camunda-ppm-hpo/.
[3] https://camunda.com/case-studies/.

2 Background

As a couple of PPM approaches appeared in the last ten years, there also exist detailed literature reviews [5,11]. [11] illustrates the importance of the research area by showing the growing amount of related publications from 2010 to 2016. Furthermore, from [5], it can be observed that neural networks are frequently used for next activity predictions. Another discovery is that the preponderance of implementations are realized with WEKA [7] or ProM-Tools[4]. Nonetheless, both articles conclude with addressing the need for practical implementations.

2.1 Predictive Process Monitoring

PPM is performed on historic data from process executions, which is usually stored in the XES log standard. An event log $\mathcal{L}$ is a collection of traces $\tau_{0...n}$ from distinct process instances, i.e., possible traversals through a process model. Each trace τ consists of events $e_{1...m}$, such as conducting an activity or receiving a message. Moreover, an event can store additional properties like the assigned employee. In order to make predictions on an event log, the log data has to be encoded to subsequently train a predictive model. Then, depending on its configuration and capabilities, the classifier returns a prediction such as the probability of a task to occur next, the remaining time or the risk to fail. Taylored solutions may also include linear temporal logic business rules [10], alarm based risk models [12] or additional preprocessing steps like clustering [4]. For the actual training process, it is common to split the available data into a training and a validation set. Hence, one can train the classifier on the training set and can calculate a quality metric or a loss function for the validation set.

When training a model, the training process itself often depends on hyper parameters. For instance, neural networks require a variety of parameters such as the number of epochs. Research in machine learning has shown that the outcome of a prediction algorithm may largely depend on its configuration and the underlying data set. Therefore, parameter optimization is an important challenge in the scope of achieving more accurate predictions.

2.2 Hyper Parameter Optimization

The configuration space for a machine learning algorithm $\mathcal{A}$ with N hyper parameters is denoted as $\Lambda = \Lambda_1 \times \Lambda_2 ... \Lambda_N$ where Λ_n is the domain of the n-th hyper parameter. The domain can be real-valued, ordinal, binary or categorical. An instance of $\mathcal{A}$ to a vector of hyper parameters $\lambda \in \Lambda$ is written as $\mathcal{A}_\lambda$. For a given data set D, the optimization problem can be formulated as:

$$\lambda^* = \text{argmin}_{\lambda \in \Lambda} \mathbb{E}_{(\mathcal{D}_{train}, \mathcal{D}_{valid} \sim \mathcal{D})} \mathbf{V}(L, \mathcal{A}_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid}) \tag{1}$$

where $\mathbf{V}(L, \mathcal{A}_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid})$ is the loss the algorithm $\mathcal{A}_\lambda$ trained on $\mathcal{D}_{train}$ and validated on $\mathcal{D}_{valid}$. Typically used loss functions are sum of squared error

[4] https://www.promtools.org/doku.php.

or misclassification rate. Vice versa, a quality metric like accuracy can be maximized. For conducting an actual optimization, different strategies such as Grid Search, population-based strategies or bayesian optimization can be applied [6].

Grid Search is the most trivial way to find the minimum of the objective function. It evaluates all permutations of configurations on a finite configuration space and returns the best one. The disadvantage of this simple approach is that the number of executions and the runtime consequently increases dramatically when being applied on a large configuration space. Random Search enhances Grid Search by traversing the configuration space in a random manner for a fixed number of iterations. By design, important parameters are detected faster and evaluated more densely [1].

Population-based strategies like evolutionary algorithms are initialized with a random set of configurations. Thereupon, genetic operations such as crossover, mutation and elitism are applied to generate a new set of configurations. A fixed number of generations or a saturating loss rate forces the algorithm to terminate. The most popular genetic algorithm for HPO is CMA-ES [8], whereas a current genetic algorithm for the PPM domain can be found in [3].

Bayesian Optimization tackles the problem that a configuration space may become infinitely large by creating a model that predicts the best possible configuration. Initially, the model is build on a randomly selected set of configurations considering the observed loss rates. In the following step, the model predicts a configuration that will have a low expected loss rate. Eventually the actual loss rate is evaluated and used to refine the model. The last two steps are replayed iteratively until the loss rate saturates or a maximal number of iterations is reached. The model used for this sequential model based optimization method can be a Gaussian Process [9] or a Tree Parzen Algorithms [2] among others.

A remarkable work in the field of HPO is Auto-WEKA [13], which enables combined selection and HPO for classification algorithms (CASH). Hence, the library is able to find the optimal algorithm with least possible user interaction. Apart from that, the resulting configuration space grows exponentially as WEKA provides 27 base classifier, 10 booster and 2 ensemble learning techniques.

As motivated, HPO is an important aspect in the context of PPM. In the following, we present our developed plugin for applied PPM, allowing for a seamless integration of results from PPM and intuitive HPO in Camunda.

3 Tool Description and Demonstration

The presented plugin is based on a previous plugin (See footnote 2), which equips Camunda with PPM functionalities. While empowering companies with means for PPM may seem beneficial, yet, creating value from a productive plugin for PPM remains a tightrope walk between complexity and usability, as some classifiers depend on a multitude of hyper parameters. Withal, it cannot be assumed that a typical Camunda user disposes of expert knowledge about HPO. Thus, the presented plugin integrates a much needed intuitive HPO functionality to create suitable predictive models.

A new plugin tab in Camunda allows to start a new parameter optimization process, with the goal of finding a) an optimal learning algorithm, and b) the best/suitable parameters for a given process model and the corresponding internal Camunda log. In this way, end-users can be supported in determining optimal algorithms and settings for model training, directly in Camunda.

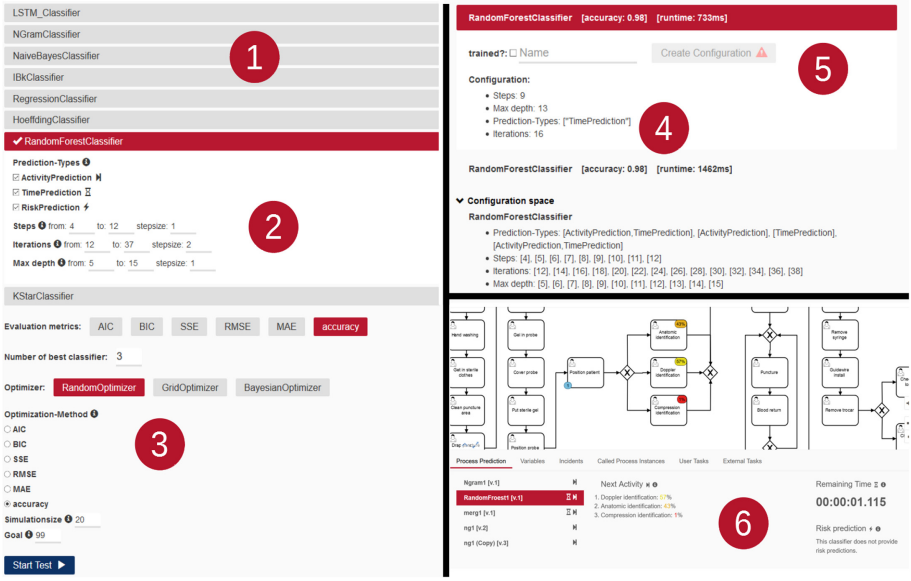

Fig. 1. Steps to find and create the best classifier. Left: Setting configuration space and optimizer. Top right: Selecting and creating a resulting classifier. Bottom right: Prediction in the process view.

Figure 1 shows the user workflow for configuring a new search and deploying the resulting classifier. Different classifier types are included in the search, allowing to compare the suitability of different learning algorithms (1). Note that arbitrary classifiers can be added if needed. Then, for each classifier type, certain ranges for the respective parameters can be specified as a general search space (2). Reasonable ranges for the hyper parameter configurations are provided as a default so that an inexperienced user only has to select which classifier should be included. Regarding the subsequent optimization process, the user may choose and customize the evaluation metrics, as well as the optimization algorithm (3). If the whole configuration space should be tested, the user can use Grid Search. Otherwise, Random Search and Bayesian Optimization offer fast and reliable recommendations after only a few iterations. In the current implementation, the model for Bayesian Optimization may be one of Random Forest, Regression or Naive Bayes. Besides that, own optimization algorithms can be added.

After the parameter optimization has been performed, the user is presented with a comprehensive overview of the best learning algorithms and the best respective parameter settings. The user can browse the n best performing configurations and select a suitable model for deployment (4). Also, the underlying parameters for the model configuration are shown. The user can directly select the "create configuration" option (5), which automatically deploys the selected predictive model in the Camunda PPM environment (6).

4 Evaluation

To investigate the feasibility of applying our plugin in an industrial setting, the runtime of the three optimization algorithms was evaluated with the help of real-life event logs from the BPI Challenge 2020[5]. This dataset includes two logs, namely the *Domestic Declarations* dataset (10.500 traces), and the *International Declarations* dataset (6.449 traces). For each dataset, scenarios with three different configuration spaces (small, medium, large) and optimization methods (Grid Optimizer, Random Optimizer, Bayesian Optimizer) were tested on five classifiers (N-Grams, IBk, Naive Bayes, Random Forest, Hoeffding Tree). A file containing the used configuration setting as well as the runtime results can be found online[6]. The experiments were performed on a CentOS Linux Server with i7-3770 CPU and 16 GB RAM.

Figure 2 shows the total runtime of the individual optimization methods. As can be seen, the Bayesian Optimizer outerperforms the other approaches significantly w.r.t. runtime. For Grid and Random Optimizer, runtime increases with growing configuration space. Moreover, Table 1 shows the runtime per classifier. Only classifier with large configuration spaces seem to be found faster by the Bayesian Optimizer. For example, the runtime of N-Grams does not differ a lot per optimization method, whereas a considerable speedup can be observed for Random Forest. Noticeably, the model accuracy while using the prediction-based Bayesian Optimizer does not seem to drop significantly as opposed to a brute-force approach via the Grid Optimizer. Thus, based on our results, the Bayesian Optimizer can be recommended, especially w.r.t. runtime.

Because of a lack of computational power, we were not able to include Regression and LSTM classifiers in our evaluation. Especially for the LSTM with its large configuration space and its massive training effort, it would be interesting to determine the computational speedup. As the capabilities of some classifier increase, difficulties arise when one wants to conduct a comparison objectively. Accordingly, further tests may include larger training logs and more sophisticated prediction types such as risk models.

[5] https://data.4tu.nl/collections/BPI_Challenge_2020/5065541.
[6] https://gitlab.uni-koblenz.de/fg-bks/camunda-ppm-hpo/blob/master/Resources/tables.pdf.

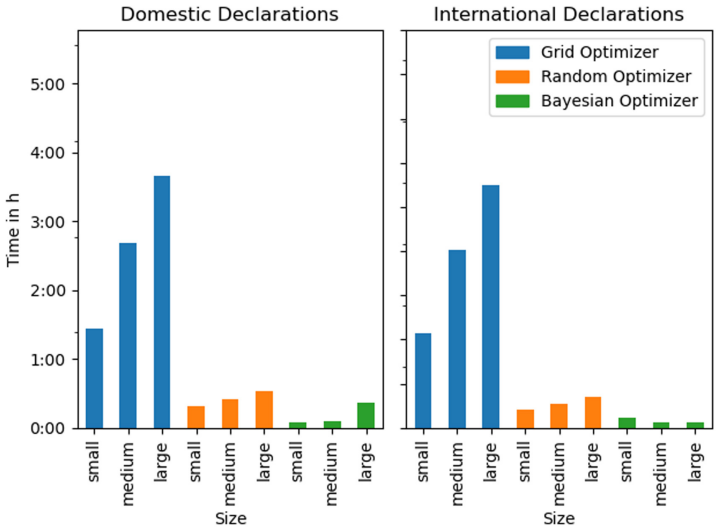

Fig. 2. Runtime evaluation for grouped by dataset, optimizer and size of configuration space.

Table 1. Evaluation of the runtime per classifier. (D: Domestic Declarations, I: International Declarations, G: Grid Optimizer, R: Random Optimizer, B: Bayesian Optimizer, each cell contains runtime in seconds and accuracy in percent)

Data	Opt.	Size	N-Grams	IBk	Naive Bayes	Random Forest	Hoeffding
D	G	Small	.452 s - 98%	685.8 s - 97%	7.79 s - 86%	3705 s - 97%	776.1 s - 91%
D	G	Medium	.565 s - 98%	1042 s - 97%	11.47 s - 87%	7393 s - 97%	1225 s - 91%
D	G	Large	.834 s - 98%	1411 s - 97%	15.07 s - 91%	0272 s - 97%	1495 s - 91%
D	R	Small	.412 s - 98%	667.3 s - 97%	7.462 s - 86%	373.8 s - 97%	83.97 s - 91%
D	R	Medium	.585 s - 98%	1042 s - 97%	11.64 s - 87%	371.1 s - 97%	83.52 s - 91%
D	R	Large	.839 s - 98%	1415 s - 97%	15.58 s - 91%	395.6 s - 97%	83.80 s - 91%
D	B	Small	.408 s - 98%	259.6 s - 97%	7.54 s - 86%	11.20 s - 97%	3.178 s - 91%
D	B	Medium	.568 s - 98%	338.6 s - 97%	3.51 s - 87%	12.87 s - 97%	5.494 s - 91%
D	B	Large	.848 s - 98%	1282 s - 97%	3.85 s - 86%	44.02 s - 97%	3.539 s - 91%
I	G	Small	.634 s - 98%	747.3 s - 96%	11.18 s - 82%	5958 s - 96%	984.2 s - 86%
I	G	Medium	.917 s - 98%	1246 s - 96%	17.53 s - 82%	1702 s - 96%	1557.4 s - 87%
I	G	Large	1.47 s - 98%	168.2 s - 96%	23.03 s - 82%	6218 s - 96%	1881 s - 87%
I	R	Small	.683 s - 98%	762.4 s - 96%	11.11 s - 82%	593.7 s - 96%	108.5 s - 86%
I	R	Medium	.939 s - 98%	1216 s - 96%	17.26 s - 82%	615.9 s - 96%	108.4 s - 87%
I	R	Large	1.47 s - 98%	172.7 s - 96%	22.80 s - 82%	638.5 s - 96%	106.6 s - 87%
I	B	Small	.644 s - 98%	736.3 s - 96%	11.19 s - 82%	21.61 s - 96%	8.46 s - 86%
I	B	Medium	.842 s - 98%	436.9 s - 96%	6.63 s - 82%	15.39 s - 96%	6.44 s - 86%
I	B	Large	1.52 s - 98%	408.9 s - 96%	4.62 s - 82%	18.06 s - 96%	5.25 s - 86%

5 Conclusion

The field of PPM is still developing and solutions for productive environments are still pending. With the introduced plugin, users can effortlessly create powerful and suitable predictive models for immediate predictions, without the need for

extensive knowledge on the underlying machine learning algorithms. The direct integration into Camunda promotes a seamless application of predictive process monitoring, without the need for isolated tools and (redundant) data transferals between tools.

Further challenges include the development of additional prediction types like linear temporal logic and further evaluation metrics like Cohen's Kappa. Here again, the trade-off between usability and degrees of freedom must be balanced. Besides, in situations where only little or no training data is available, the results of the HPO are overfitted and thus not meaningful. An integrated simulation environment to create synthetic logs would be useful in these cases.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(1), 281–305 (2012)
2. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554 (2011)
3. Di Francescomarino, C., et al.: Genetic algorithms for hyperparameter optimization in predictive business process monitoring. Inf. Syst. **74**, 67–83 (2018)
4. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. IEEE Trans. Serv. Comput. **12**, 896–909 (2016)
5. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F.: Predictive process monitoring methods: which one suits me best? In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 462–479. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_27
6. Feurer, M., Hutter, F.: Hyperparameter optimization. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning. TSSCML, pp. 3–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_1
7. Frank, E., Hall, M.A., Witten, I.H.: The WEKA Workbench. M. Kaufmann (2016)
8. Friedrichs, F., Igel, C.: Evolutionary tuning of multiple SVM parameters. Neurocomputing **64**, 107–117 (2005)
9. Hutter, F., Lücke, J., Schmidt-Thieme, L.: Beyond manual tuning of hyperparameters. KI-Künstliche Intelligenz **29**(4), 329–337 (2015)
10. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., et al. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_31
11. Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortes, A.: Predictive monitoring of business processes: a survey. IEEE Trans. Serv. Comput. **11**(6), 962–977 (2017)
12. Teinemaa, I., Tax, N., de Leoni, M., Dumas, M., Maggi, F.M.: Alarm-based prescriptive process monitoring. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNBIP, vol. 329, pp. 91–107. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98651-7_6
13. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 847–855 (2013)