A Scikit-learn Extension Dedicated to Process Mining Purposes (Extended Abstract)

Rafael Seidi Oyamada[1,*], Gabriel Marques Tavares[2], Sylvio Barbon Junior[3] and Paolo Ceravolo[1]

[1]*University of Milan*
[2]*LMU Munich*
[3]*University of Trieste*

Abstract
This demo paper presents an extension of the Scikit-learn library tailored for Process Mining applications. By integrating Process Mining capabilities into the well-known Scikit-learn, this work enables the establishment of standardized preprocessing procedures and learning workflows for researchers and practitioners. The significance of this library stems from the unaddressed challenge of reproducibility within the Process Mining community, coupled with the absence of benchmarking resources. The library is publicly accessible on GitHub, facilitating widespread adoption and collaboration in the field: https://github.com/raseidi/skpm.

Keywords
Process Mining, Scikit-learn, Machine Learning, Predictive Monitoring

1. Introduction

In the realm of process mining (PM), a research discipline characterized by its unique reliance on event data, the absence of a standardized library for conducting machine learning (ML) experiments has been a persistent challenge. This lack of dedicated tooling has given rise to a problem of inefficiency and reproducibility limitations. Given the distinctive nature of process data, conventional ML libraries often fail to provide the requisite flexibility and functionality required for effective experimentation in this domain.

The primary objective of this paper is to address this longstanding issue by proposing an extension to the widely recognized Scikit-learn library tailored specifically to the demands of process mining. Scikit-learn, renowned for its versatility and user-friendly application programming interface (API), has emerged as a foundation in the ML community, empowering researchers to construct intricate pipelines, ensuring reproducibility, and facilitating comprehensive benchmarking.

Proceedings of the Demonstration Track at International Conference on Cooperative Information Systems 2023, CoopIS 2023, Groningen, The Netherlands, October 30 - November 3, 2023
*Corresponding author.
rafael.oyamada@unimi.it (R. S. Oyamada); tavares@dbs.ifi.lmu.de (G. M. Tavares); sylvio.barbonjunior@units.it (S. B. Junior); paolo.ceravolo@unimi.it (P. Ceravolo)
0000-0002-2601-8108 (G. M. Tavares); 0000-0002-4988-0702 (S. B. Junior); 0000-0002-4519-0173 (P. Ceravolo)

In light of this motivation, our endeavor seeks to bridge the gap between the demands of process mining and the capabilities of a mature and well-established ML library. In doing so, we aim to enhance the accessibility and efficiency of ML experiments within the PM domain, ultimately contributing to the advancement of this research field. In the subsequent sections of this paper, we will delve into the specifics of our proposed extension, describe its design and functionalities, and present a few use cases for predicting the remaining time and the next activity of processes.

The paper is organized as follows: we motivate and discuss existing tools in Section 2, describe the proposed implementation in Section 3 followed by a brief use can in Section 4. We conclude and discuss future research in Section 5.

2. Tool Motivation

Within the domain of PM, where the unique characteristics of the data demand specialized attention, there exists a need for a dedicated ML library to facilitate the essential aspects of reproducibility and benchmarking. The data usually consists of an event log, which is a set of process instances (cases/traces), where a process instance is an ordered sequence of events. Each event is represented by at least a case identifier (it identifies to which process instance the event belongs), a timestamp (remarking when the event was executed), and an activity label (remarking the action that represents the event, for instance, *Create process* or *Conclude process*). This nature of data, which is commonly collected from large information systems, requires specific preprocessing steps for ML purposes which usually consist of extracting temporal feature representations, mapping resource usage, and measuring process costs [1].

Existing tools, including generic PM software (e.g., ProM[1] [2], Apromore[2] [3], and pm4py[3] [4]), process analytics tools (e.g., DyLo[4] [5]), and comprehensive surveys offering open-source access to their experimental setups [6, 7], while valuable resources, do not fulfill the role of dedicated tools for practical ML application and extensibility. The benchmarks commonly employed are static in nature and present challenges when it comes to extensibility and adaptation. The well-known *pm4py* may be considered the most popular tool for managing event logs. Although the library has a module dedicated to ML, it is limited to a few simple feature engineering/extraction solutions. Thus, in the end, the design of ML pipelines consisting of tuning for both preprocessing and training steps is not supported, i.e., is left to the user. Our innovative proposal streamlines ML pipelines for PM, coupling all necessary steps in a single tool, reducing practitioner effort.

Within process mining, a popular field that employs learned solutions is called Predictive Process Monitoring [6, 7] and it had a huge increase of publications in the past few years. The motivation for our work becomes evident when considering the existing benchmark practices. While these benchmarks utilize the core methods of *Scikit-learn* and *pm4py*, the incorporation of essential elements for constructing ML pipelines tailored to PM objectives necessitates significant overhead. This eventually results in the reliance on external libraries beyond Scikit-

[1]https://promtools.org/
[2]https://apromore.com/
[3]https://pm4py.fit.fraunhofer.de/static/assets/api/2.7.5.1/index.html
[4]https://github.com/BrechtWts/DyLoPro

learn, introducing complexities in terms of maintenance, reproducibility, and adaptability as these external dependencies may quickly become outdated.

3. Design and Implementation

This section presents technical details. First, we present an overview of how to extend Scikit-learn. Second, we discuss how PM-related features/steps might be included under the Scikit-learn API.

Scikit-learn Overview. In summary, the Scikit-learn API involves the following fundamental steps: firstly, the *instantiation* of an estimator or transformer; secondly, the *fitting* of the object instance to acquire knowledge from the input data set; and finally, the *prediction* process if the object serves as a predictor, such as a random forest, or the *transformation* process if the object is intended to modify the data in a certain manner. From the official documentation, we summarize the main characteristics of the library below.

- **Instantiation**. The `__init__()` should contain arguments that determine the estimator's behavior. More specifically, only tunable hyperparameters should be included here. This is important in more complex applications, for instance when performing grid search, to differentiate the hyperparameters from attributes. No logic nor validation should be implemented here.
- **Fitting**. The `fit()` method takes the training data as an argument, which can be just one array if it is an unsupervised problem or a data transformation; or two arrays if it is a supervised learning scenario. Additionally, it accepts optional data-dependent parameters.
- **Predicting or Transforming**. If `fit()` succeeds, it will estimate parameters from the data (e.g., the mean and standard deviation for z-score normalization). Finally, the methods `predict()` or `transform()` can be called, depending on the nature of the estimator.

Extending Scikit-learn for Process Mining. Following the specified criteria, we formulate a Scikit-learn extension for PM by utilizing its core methods for standardizing ML experiments in the PM community. In this initial work, our focus centers on the event attributes, including the case identifier, timestamp, activity, and categorical resources.

For activity representation, we employ one-hot encoding, aligning with the prevalent practice in the PM community, as indicated by prior research [6], despite recent findings suggesting superior performance with alternative encoding methods [8]. Concerning timestamp-related feature extraction, our library accounts for both the case identifier and the timestamp to ensure proper attribution of events to their respective process instances, especially when calculating time-based metrics such as the time difference between consecutive events (e_{i-1} and e_i). Additionally, while categorical resource attributes (e.g. actors playing events) can be one-hot encoded, our library also offers an alternative organizational mining algorithm for role discovery [9]. This algorithm identifies common roles among actors, facilitating resource grouping and reducing category dimensionality.

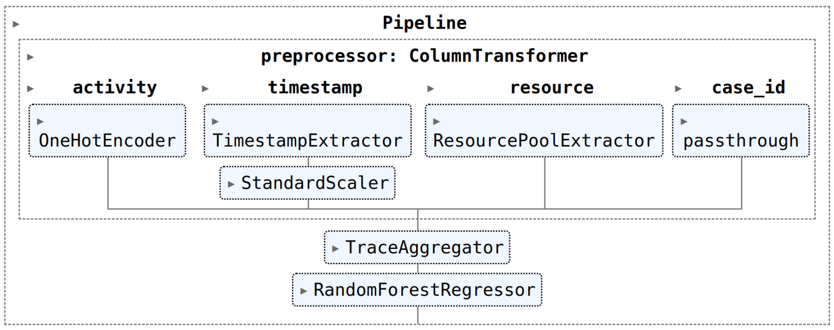

Figure 1: An example of our SkPM pipeline implemented extending the traditional Scikit-learn library.

When instantiating our estimators, users have the flexibility to specify parameters such as the time-related features to extract from timestamps and the sensitivity threshold for the role discovery algorithm. During the fit method invocation, users must provide the column names corresponding to each required event attribute for a given estimator. By meticulously designing each estimator for event feature extraction, we seamlessly integrate them into the well-established Scikit-learn pipelines and model selection algorithms. In the subsequent section, we illustrate a practical use case by constructing a concise pipeline using our proposed library.

4. Use cases

Our work is under continuous development and we are already able to perform a complete pipeline involving preprocessing and prediction. In this section, we introduce a pipeline for remaining time prediction and demonstrate it in a video record[5].

We have implemented two feature extractors, one for timestamps and another for categorical resources, and tested them on public event logs (see our repository). The class `TimestampExtractor` has only one argument `features` that describes all the features that should be extracted from the timestamp. These time-related features were collected from the literature and they currently include `execution_time`, `accumulated_time`, `within_day`, and `remaining_time`. The latter is commonly employed as a target to be predicted. Regarding the categorical resources, we have implemented the resource role discovery as mentioned in the previous section, which in a nutshell aggregates common resources into groups (roles in the organization), hence reducing the set of available categorical values. This algorithm has a hyperparameter that is used as a relationship threshold, which can be seen as a threshold to decide if resources belong or not to the same group.

That said, we can instantiate a Scikit-learn pipeline as illustrated in Figure1. For activity encoding, we simply use the traditional one-hot encoding. The timestamp can have all the implemented features extracted in order to increase the knowledge space, and normalized subsequently. The resource has its set of unique and individual labels reduced to roles (a.k.a. pool). Lastly, the case identifier needs to be passed through, i.e., it is fed to the ColumnTransformer and

[5]https://tinyurl.com/2hvt3zfk

should not be dropped (default operation in Scikit-learn) in order to be used in the next step. The TraceAggregator is a trace encoding technique described in [6] which consists of aggregating the previous events of an ongoing case in order to output a single array. This aggregation considers averaging or summing all the knowledge up. Finally, a random forest or any other learning algorithm can be employed. In our example, we illustrate the *RandomForestRegressor* since we are predicting the remaining time of ongoing cases. The reader is able to run the pipeline by following the instructions in our repository.

5. Conclusion and discussion

In this paper, we present an initial iteration of SkPM, a specialized ML library tailored for the PM domain. This library focuses on the implementation of widely adopted preprocessing steps from PM involving ML-based solutions. SkPM capitalizes on the full spectrum of functionalities provided by the renowned Scikit-learn framework, encompassing standardization of preprocessing steps, the construction of pipelines, and the execution of model selection procedures. Ongoing efforts are dedicated to the continual extension of SkPM's features, enhancements to its usability, refinement of documentation, and the provision of tutorials. Lastly, we invite researchers and practitioners to actively participate and contribute to SkPM's development by visiting our repository at https://github.com/raseidi/skpm.

References

[1] C. dos Santos Garcia, A. Meincheim, E. R. F. Junior, M. R. Dallagassa, D. M. V. Sato, D. R. Carvalho, E. A. P. Santos, E. E. Scalabrin, Process mining techniques and applications - A systematic mapping study, Expert Syst. Appl. 133 (2019) 260–295.

[2] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, W. M. P. van der Aalst, The prom framework: A new era in process mining tool support, in: ICATPN, volume 3536 of *LNCS*, Springer, 2005, pp. 444–454.

[3] M. L. Rosa, H. A. Reijers, W. M. P. van der Aalst, R. M. Dijkman, J. Mendling, M. Dumas, L. García-Bañuelos, APROMORE: an advanced process model repository, Expert Syst. Appl. 38 (2011) 7029–7040.

[4] A. Berti, S. van Zelst, D. Schuster, Pm4py: A process mining library for python, Software Impacts 17 (2023) 100556.

[5] B. Wuyts, H. Weytjens, S. vanden Broucke, J. D. Weerdt, Dylopro: Profiling the dynamics of event logs, in: BPM, volume 14159 of *LNCS*, Springer, 2023, pp. 146–162.

[6] I. Teinemaa, M. Dumas, M. L. Rosa, F. M. Maggi, Outcome-oriented predictive process monitoring: Review and benchmark, ACM Trans. Knowl. Discov. Data 13 (2019) 17:1–17:57.

[7] E. Rama-Maneiro, J. Vidal, M. Lama, Deep learning for predictive business process monitoring: Review and benchmark, IEEE TSC (2021) 1–1.

[8] S. B. Jr., P. Ceravolo, R. S. Oyamada, G. M. Tavares, Trace encoding in process mining: a survey and benchmarking, CoRR abs/2301.02167 (2023).

[9] M. Song, W. M. P. van der Aalst, Towards comprehensive support for organizational mining, Decis. Support Syst. 46 (2008) 300–317.