Design and Evaluation of a Process-aware Recommender System based on Prescriptive Analytics

Massimiliano de Leoni
University of Padua
Padua, Italy
deleoni@math.unipd.it

Marcus Dees
UWV & Eindhoven University of Technology
Amsterdam & Eindhoven, The Netherlands
Marcus.Dees@uwv.nl

Laurens Reulink
CZ
Tilburg, The Netherlands
laurensreulink@gmail.com

***Abstract*—Process-aware Recommender systems (PAR systems) are information systems that aim to monitor process executions, predict their outcome, and recommend effective interventions to reduce the risk of failure. While a PAR system is composed by monitoring, predictive analytics and prescriptive analytics, the lion's share of attention in the recent years has been on the first two, overlooking the last. It seems that process participants are tacitly assumed to take the "right decision" for the most appropriate corrective actions in case of failure's risks. Unfortunately, the assumption of selecting an effective corrective action is not always met in reality. When selecting an intervention, this is mainly based on human judgment, which naturally relies on subjective process' perceptions, instead of objective facts. Experience has shown that, when a fact-based predictive analytics is followed by subjective prescriptive analytics, the positive effect of good predictions are nullified by inconclusive corrective actions, yielding no final improvement. This paper discusses a PAR system that features a data-driven prescriptive analytics framework, which puts aside subjective options and focuses on factual data. The effectiveness of the proposed solution is assessed through the process of a reintegration company, showing a potential increase of customers that find a new job.**

***Index Terms*—Prescriptive Analytics, Recommender Systems, Process Improvement, Machine Learning, Transition Systems**

I. Introduction

Process-aware Recommender systems (hereafter shortened as PAR systems) are a specific class of Information Systems that aim to predict how the executions of process instances are going to evolve in the future, to determine those that have higher chances to not meet desired levels of performance (e.g. costs, deadlines, customer satisfaction) and, consequently, to provide recommendations on which contingency actions should be enacted to try to recover the risky executions. PAR systems are hence expert systems that run in background and continuously monitor the executions, predicting their future and, possibly, recommend intervention actions.

Conceptually, a PAR system is composed by three subsystems/blocks: *(i)* monitoring, which keeps tracks of running process instances, *(ii)* a predictive-analytics block, which forecasts the future outcome of running instances, and *(iii)* a prescriptive-analytics system, which provides recommendations on the running instances that risk to conclude with a poor outcome.

While a large body of research has been conducted on monitoring and predictive analytics, little attention has been paid on generating recommendations (cf. Section V). Process participants are somehow implicitly assumed to take the "right decision" for the most appropriate corrective actions for each case. Unfortunately, the assumption of selecting an effective corrective action is not always met in reality. When selecting an intervention, this is mainly done based on human judgment, which naturally relies on the subjective perception of the process instead of being based on objective facts. In [1], the authors developed a predictive-analytics module that built on machine-learning techniques, and rely on historical data, and discussed potential interventions with process stakeholders. A subsequent field experiment, with real process instances and a selected intervention, showed that, while process instances were predicted rather well, the intervention did not have the desired effect. No significant improvement of the average outcome was observed, even when enacting the selected intervention. The final lesson was that *accurate predictions are crucial, but their effect is nullified if it is not matched by effective recommendations, and effective recommendations must be based on objective evidence from historical process data*.

This paper reports on the design of a prescriptive-analytics technique that recommends which actions/activities to perform next to optimize a certain KPI (Key Performance Indicator) of interest. The recommendations put aside the human subjectivity and rely on the process' transactional data, recorded in the so-called event logs. This way, the recommendations are purely objective, and not biased.

In a nutshell, our prescriptive-analytics proposal relies on a predictive-analytics module. For each running case with predicted, poor KPI values, we first simulate any possible continuation of the running case, i.e. we simulate the situation in which each possible activity is performed as next. Then, we predict the final KPI value for each continuation. Finally, the system recommends the activity (activities) that is (are) predicted to largely improve the KPI value. Of course, one should only simulate those continuations (i.e. next activities) that are meaningful from a domain viewpoint. If one had a

business process model, this would be easy. However, the presence of a process model is a rather strong assumption. Therefore, an event log of complete process instances is used as input to build a transition system that abstracts the observed behavior. Running cases are mapped onto states of this transition system: the meaningful next activities correspond to the transitions enabled at those states.

The three modules (monitoring, predictive and prescriptive analytics) have been implemented as a PAR system in Python. The entire system was assessed on real-life process data of a reintegration company, aiming to maximize the percentage of customers finding a new job (the KPI). The assessment results showed that the use of the PAR system would significantly reduce the percentage of unsuccessful process instances. The statistical significance of the results was also verified, thus providing a sound, successful validation of the quality of our prescriptive analytics.

Section II introduces the basic concept on which our prescriptive analytics builds: event logs, KPIs, and predictive analytics. Section III reports on the design of our framework for prescriptive analytics, while Section IV discusses the implementation and the experiments. Section V compares our prescriptive analytics with the state of the art. Finally, Section VI concludes this paper, summarizing the contribution, the lessons learnt, and the future research directions.

II. Preliminaries

Section II-A introduces the starting point of our process-aware recommender system based on predictive and prescriptive analytics: the event logs. Section II-B formalizes the concept of KPI, while Section II-C introduces preliminary concepts of predictive analytics.

A. Event Logs

The starting point for a prediction system is an *event log*. An event log is a multiset of *traces*. Each trace describes the life-cycle of a particular *process instance* (i.e., a *case*) in terms of the *activities* executed and the process *attributes* that are manipulated.

Definition 1 (Events): Let $\mathcal{A}$ be the set of process' activities. Let $\mathcal{V}$ be the set of process attributes. Let $\mathcal{W}_{\mathcal{V}}$ be a function that assigns a domain $\mathcal{W}_{\mathcal{V}}(x)$ to each process attribute $x \in \mathcal{V}$. Let $\overline{\mathcal{W}} = \cup_{x \in \mathcal{V}} \mathcal{W}_{\mathcal{V}}(x)$. An event is a pair $(a, v) \in \mathcal{A} \times (\mathcal{V} \not\to \overline{\mathcal{W}})$ where a is the event activity and v is a partial function assigning values to process attributes, with $v(x) \in \mathcal{W}_{\mathcal{V}}(x)$.
A trace is a sequence of events. Note that the same event can potentially occur in different traces, namely attributes are given the same assignment in different traces. This means that potentially the entire same trace can appear multiple times. This motivates why an event log is to be defined as a multiset of traces:[1]

Definition 2 (Traces & Event Logs): Let $\mathcal{E} \subset \mathcal{A} \times (\mathcal{V} \not\to \overline{\mathcal{W}})$ be the universe of events. A trace σ is a sequence of events, i.e. $\sigma \in \mathcal{E}^*$. An event-log L is a multiset of traces, i.e. $L \subset \mathbb{B}(\mathcal{E}^*)$.

[1]Given a set X, $\mathbb{B}(X)$ indicates the set of all multisets with the elements in X.

Given an event $e = (a, v)$, the remainder uses the following shortcuts: $activity(e) = a$ and $variables(e) = v$. Also, given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$, $prefix(\sigma)$ denotes the set of all σ's prefixes: $\{\langle\rangle, \langle e_1 \rangle, \langle e_1, e_2 \rangle, \ldots, \langle e_1, \ldots, e_n \rangle\}$.

B. Key Performance Indicators

A PAR system aims to optimize the Key Performance Indicators (KPIs) of processes. KPIs can be of any nature:

Definition 3 (KPI Function): Let $\mathcal{E}^*$ be the universe of events defined over a set $\mathcal{V}$ of attributes. A KPI is a function $\mathcal{T} : \mathcal{E}^* \to \mathbb{R}$ such that, given a complete trace $\sigma \in \mathcal{E}^*$, $\mathcal{T}(\sigma)$ returns the KPI value of σ.

Given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$ that records a complete process execution, the following are four potential KPI definitions:

- **Duration.** $\mathcal{T}_{duration}(\sigma)$ is equal to the difference between the timestamp of e_n and that of e_1.
- **Activity Occurrence.** It measures whether a certain activity is observed in the trace, such as an activity *Open Loan* in a loan-application process. The corresponding KPI definition for the occurrence of an activity A is $\mathcal{T}_{occur\_A}(\sigma)$, which is equal to the number of events in σ that refers to A.
- **Customer Satisfaction.** This is typical KPI to analyze for processes related to service provision. Let us assume, without losing generality, to have a trace $\sigma = \langle e_1, \ldots, e_n \rangle$ where the satisfaction is known at the end, e.g. through a questionnaire. Assuming the satisfaction level is recorded with the last event - say $e_n(sat)$. Then, $\mathcal{T}_{cust\_satisf}(\sigma) = variables(e_n)(sat)$.

C. Predictive Analytics

Predictive Analytics aims to forecast the execution and/or the outcome of a running case. A large share of attention in Process Mining has been devoted to business process predictions (cf. Section V). Within our recommender-system framework for KPI optimization, the predictive analytics block can be defined as follows, specializing the definition by Senderovich et al. [2]. Given a running case $\sigma_{run} = \langle e_1, \ldots, e_k \rangle$, which will eventually end with a trace $\sigma = \langle e_1, \ldots, e_k, e_{k+1}, \ldots, e_n \rangle$, predictive analytics can be abstracted as learning a *predictive-analytics oracle* $\mathcal{P}(\sigma_{run})$ that forecasts the KPI value $\mathcal{T}(\sigma)$. It follows that a perfect prediction is such that $\mathcal{P}(\sigma_{run}) = \mathcal{T}(\sigma_T)$.

Predictive-analytics oracles can be built in several ways, and using several machine-learning approaches. It requires a training event log L_T. A typical predictive-analytics block of a recommender system follows three phases as illustrated in Fig. 1 (cf. [3], [4]):

- **Extract Prefixes.** We build the multiset of all prefixes, which is associated with a corresponding KPI value: $\uplus_{\sigma \in L_T} \uplus_{\sigma' \in prefix(\sigma)} (\sigma', \mathcal{T}(\sigma))$. [2]
- **Encode Prefixes for Training.** The predictor is trained on observation instances (d, i) where d is the vector that

[2]Symbol $\uplus$ indicates the multiset union, namely duplicates are retained.

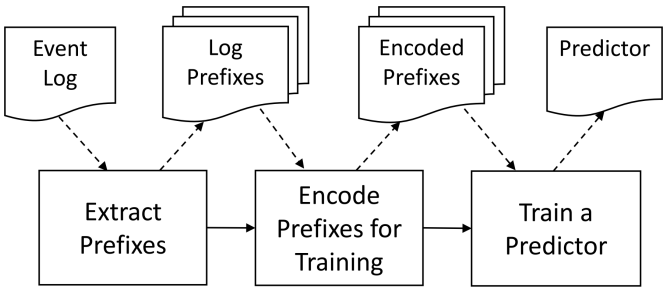


Fig. 1: The training of the predictive model.

encodes the independent variables and i is the dependent variable. The pairs of prefixes and KPI values (σ', κ) need to be converted into an observation instance (d, i) where $i = \kappa$ and d is the encoding of σ'. This encoding can be achieved using consolidated techniques [3], [4]. In particular, Leontjeva et al. show that a frequency-vector encoding is a good balance between abstraction richness and complexity: there is one vector's element per process' activity $a \in \mathcal{A}$, and there is one element per process' attribute $v \in \mathcal{V}$. The value of the element for activity a is equal to number of occurrence of a in σ'; the value for the element of variable v is equal to the value of the latest variable assignment of v.

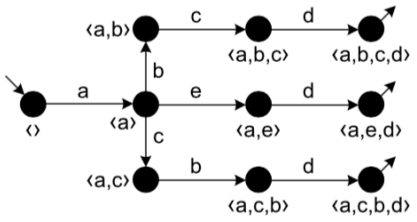
- **Train a Predictor.** The set of encoded prefixes (with the associated KPI values) are the input to train the predictor. A common case is to have prefixes with dozens of different activities and variables, which are encoded via vectors with many elements (see, e.g., the data set reported in Section IV, and used for evaluation). This poses risks of over-fitting due the problem of "curse of dimensionality" [5]. It is thus crucial to perform a *feature selection* to reduce the number of elements [5]. Note that this also reduces the model complexity, and hence it can be learnt and used faster.

III. A Framework for Prescriptive Analytics

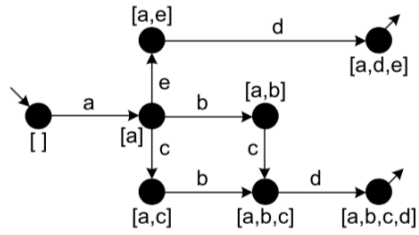
Our prescriptive-analytics framework is based on the availability of a model that was learnt using any of the predictive analytics approaches available in literature (cf. Section V). As indicated in Section II-C, we abstracted from the specific model and approach, assuming that this is exposed as a predictive-analytics function $\mathcal{P}(\sigma)$.

A. Transition System Abstractions of Event Logs

In a nutshell, given a running case, our framework considers all potential ways to continue the case execution, and predicts the risk. However, we do not assume here that process stakeholders need to draw a process model that prescribes how to continue the case executions. This might require a considerable amount of work, and the resulting model might not accurately represent the executions observed in the reality, thus suggesting recommendations that are not valid from a business viewpoint. Following the idea described by van der Aalst et al. [6], [7], valid recommendations are obtained by representing



(a) Using a Sequence Abstraction



(b) Using a multiset abstraction

Fig. 2: Transition systems representing an event log $L = \{\langle a, b, c, d\rangle, \langle a, e, d\rangle, \langle a, c, b, d\rangle\}$ [6]. Event's attributes are abstracted out for simplicity of explanation.

the past executions recorded in the event logs as a transition system. This requires to provide a *state-representation function* $l^{state} : \mathcal{E}^* \to R$ that, for each sequence of events σ, returns a state $l^{state}(\sigma) \in R$ that abstracts σ.

As an example, let us consider an event log $L = \{\langle a, b, c, d\rangle, \langle a, e, d\rangle, \langle a, c, b, d\rangle\}$. We are abstracting here out the event's attributes to keep the example simple. Fig. 2 illustrates two potential transition systems abstracting L: Fig. 2a makes use of a state-representation function $l^{state}(\sigma) = \sigma$, while Fig. 2b uses $l^{state}(\sigma)$ as the multiset of every activity that occurred in σ with the respective cardinality. A transition system is defined as follows:[3]

Definition 4 (Transition-System Abstraction of a Log): Let L be an event log defined over a set $\mathcal{V}$ of attributes, and a set $\mathcal{A}$ of activities. Let $l^{state} : \mathcal{E}^* \to R$ be a state-representation function. A transition system abstracting L is a pair $TS_L = (S, T) \subseteq R \times (R \times \mathcal{E} \times R)$ where

- S = $\cup_{\sigma \in L} \cup_{\sigma' \in prefix(\sigma)} l^{state}(\sigma')$, and
- $T = \{(l^{state}(\sigma'), e, l^{state}(\sigma' \oplus \langle e\rangle))$ s.t. $\exists \sigma \in L$ $\sigma' \oplus \langle e\rangle \in prefix(\sigma)\}$

It is known that transitions systems are generally not suitable to represent business processes, because all possible activity's interleavings need to be explicitly represented, making the transition systems very hard to read in the general case. However, here a transition system is valid as a model, because it is only used internally by the prescriptive-analytics module, and never shown to users.

B. Generating Recommendations

The input of the process prescriptive-analytics system is an event log L and a state representation function l^{state} (cf.

[3]Operator $\oplus$ indicates the concatenation of two sequences.

Def. 4). Log L is used to build a transition-system abstraction $TS_L = (S, T)$, based on l^{state}; L is also employed to build a prediction oracle $\mathcal{P} : \mathcal{E}^* \rightarrow \mathbb{R}$.

Let us consider a trace $\sigma \in \mathcal{E}^*$, which may or may not be part of L. As clarified later in this section, we need to find the set $minDist(TS_L, \sigma) \subseteq S$ of states at minimum distance from σ that can be computed as follows. We first compute the minimum number Υ of changes that are necessary for σ to obtain a trace $\overline{\sigma}$ s.t. $l^{state}(\overline{\sigma}) \in S$. The allowed changes are the primitive supported by the techniques for trace-to-trace alignments [8], namely the insertion and the deletion of events.

Once the minimum number Υ of changes is found, $minDist(TS_L, \sigma)$ is the set of states $l^{state}(\sigma') \in S$ s.t. σ' that is obtained from changing Υ events of σ.

With these concepts at hand, the recommendation for a running case identified by a partial trace $\sigma_{run} \in \mathcal{E}^*$ can be built as follows:

1) Find the set O of transitions (i.e. events) possible by TS_L from the states in $minDist(TS_L, \sigma_{run})$. Namely, $O = \{e \in T : \exists s \in minDist(TS_L, \sigma_{run}), \exists s' \in S.\ (s, e, s') \in T\}$.
2) Return every activity $a \in \mathcal{A}$ that maximizes or minimize $\mathcal{P}$, i.e. such that $\exists e \in O.\ activity(e) = a$, and either $\mathcal{P}(\sigma_{run} \oplus \langle e \rangle) = \max_{e' \in O} \mathcal{P}(\sigma_{run} \oplus \langle e' \rangle)$ or $\mathcal{P}(\sigma_{run} \oplus \langle e \rangle) = \min_{e' \in O} \mathcal{P}(\sigma_{run} \oplus \langle e' \rangle)$.[4]

Here, we assume that every trace in L denotes a legitimate execution of the process, and that L is sufficiently large. In this setting, all transitions enabled in each state of the transition system represent all potential recommendations that make sense from a business viewpoint. Therefore, the set O of enabled transitions at the states $l^{state}(\sigma_{run})$ are in fact activities that may be potentially recommended. Among those in O, some activities are predicted to optimize (i.e., maximize or minimize depending on the KPI) the KPI values if they are performed as the next activities. Those are the activities that the system will recommend.

In practice, activities are typically performed by humans, and their judgement is necessary to carry on process case executions. As a consequence, it might be limiting to only consider those activities that are predicted to optimize the KPI of interest, which are likely one per running case. In fact, our framework extends to the top n activities, namely the n activities whose execution is predicted to lead to the highest KPI values. Value n can be customized at run-time by users (cf. Section IV). In fact, the leitmotif of our framework is that *the recommender system should allow process actors to make informed decisions based on objective facts, but the ultimate choices must remain on the process actors, with their personal judgment*.

IV. Implementation and Evaluation

The framework discussed in Section III has been implemented in Python, and leverages on the machine-learning functionalities of the *scikit-learn* package [5]. The code base is available through GitHub [9].

The remainder of the section reports on the evaluation on a real-life case study. Section IV-A introduces the case study employed for our evaluation, while Section IV-B discusses the construction of the predictive model (i.e. the predictive-analytics oracle). Finally, Section IV-C discusses the setup of the experiments to assess the main contribution of this paper: the prescriptive-analytics technique. Finally, Section IV-D reports on the results of the experiments.

A. The Evaluation Case Study

The evaluation of our approach is performed at a Dutch reintegration company. The objective of the company is to help people who have lost their job to find a new job. Customers of the company are not always capable to find a new job by themselves. Common reasons are their age in combination with having been employed for a long time at the same employer, having been ill, economic circumstances and a mismatch between the type of work that is offered and ones capabilities. The company offers several tools like guiding hints, training, and workshops that can be deployed in an online setting or face-to-face. The company can only provide services to the customer for a limited time. The maximum duration depends on the number of years of work experience of the customer. The minimum number of months of entitlement is 3 and the maximum is 38. A customer can spend less than the maximum duration using the services, e.g., because the customer finds a new job.

The company employs hundreds of professionals that support customers to find a new job. There is a high-level prescriptive process model which prescribes two phases during the customer process. In the first phase, which relates to the first 6 months, a face-to-face contact is required. In the second phase, after the first six months, specific services, taken from the whole set of interventions, need to be offered to the customer. While the interventions that are available are the same throughout the company, it is up to the professional to decide which intervention would benefit a customer the most at each stage of the customer journey.

The initial impression of the company is that the order in which the services are offered can influence the outcome, e.g., having certain types of face-to-face meeting earlier or later in the process might be more effective. Therefore, the PAR system is expected to support the company's employee to choose the right interventions. The system harnesses the knowledge of all professionals and supports every professional in choosing the best next action for a customer.

The KPI that is used in the evaluation is: *the customer found a new job before reaching the maximum service duration*. The KPI is binary and has the value 1 when the customer found a new job before reaching the maximum duration, and 0 otherwise.

The evaluation is based on an event log L consisting of 12296 complete traces with 62 trace attributes and 302 unique

[4] We aim to maximize if higher KPI values are better than lower; otherwise, we aim to minimize.

activities.[5] The attributes refer to properties of the customers (e.g., age). The activities are the interventions by the company employees to support the customer's job seeking, as well as the actions performed by the customers to actively look for a job and to follow-up on the employee's interventions. The trace length is between 1 and 1161 events, with an average of 95 events. For the evaluation, L was divided in three parts:

- log $L_{Training}$ with 6148 traces (50%) was used to train the predictive-analytics oracle $\mathcal{P}$;
- log $L_{Testing}$ with 3074 traces was used to evaluate oracle $\mathcal{P}$, and to simulate the running traces, for which recommendations are created;
- log $L_{Similarity}$ with 3074 traces was used to evaluate the quality of the recommendations.

Under the assumption of no concept drifts, the division is done in a complete random fashion.

B. Evaluation on Predictive Analytics

The predictive analytics module is implemented in accordance with the framework discussed in Section II-C.

In order to build a predictive-analytics oracle $\mathcal{P}$, we used a two-phase approach to choose a suitable predictor model. First, we evaluated three machine-learning techniques: Random Forest, Support Vector Machine, and Decision Tree [10]. We selected the technique that scored the best. This technique was used in the second phase, where the learning parameters were tuned through hyper-parameter optimization.

In both phases, we followed the workflow discussed in Section II-C to train the models. The models were evaluated using two standard metrics: Accuracy, and AUC Score [10]. The AUC score is a valid metric here because the KPI values are not uniformly distributed: most of the executions had good KPI values, namely most of customers have eventually found a job.

Note that the evaluation excluded such Deep-Learning techniques as LSTM networks [11], because those models require a significant training time, while the models that we have employed could already score quite well (see below).

For the first phase, we employed 1000 traces of log $L_{Training}$, which generated 7827 prefixes, which were subsequently encoded. To reduce the number of data-set features, we performed a feature selection, based on *SelectKBest* method [10]. In this first, preliminary phase, Random Forest, Support Vector Machine, and Decision Tree were trained using the default parameters of the respective implementations in the SciPy package for Python.

The results are shown in Table I: Random Forests and Support Vector Machines worked better and equally well. Given the significant difference in training time, we finally opted for Random Forest.

During the second training phase, we employed all traces of log $L_{Training}$, and we performed a model training with hyper-parameter optimization, varying the number of decision trees within the forest, and the number of features to consider when split decision-tree nodes. The best model was obtained with 1000 decision trees, and four decision-tree nodes. To complete the assessment, we used the second log $L_{Testing}$ to measure AUC and accuracy. The best model scored quite well: AUC and accuracy were 0.8145 and 0.8775, respectively.

[5]For reasons of confidentiality, we are not allowed to share the event-log dataset.

Classifier	Accuracy	AUC	Training Time
Decision Tree (DT)	0.657	0.610	**31 ms**
Random Forest (RF)	**0.731**	**0.792**	107 ms
Support Vector Machine (SVC)	0.725	0.734	2580 ms

TABLE I: Metrics Score for the Models Trained by Three Machine-Learning Predictors.

C. Evaluation on Prescriptive Analytics: The Experiment Setup

Here, the prescriptive-analytics module operationalizes the technique discussed in Section III to provide recommendation. The technique builds on a predictive-analytics oracle but remains independent from any specific machine-learning technique employed.

The remainder of this section reports on the evaluation conducted using the oracle that was constructed as illustrated in Section IV-B.

In our experiments, we simulated running cases by considering the 3074 traces in $L_{Testing}$. For each trace $\sigma \in L_{Testing}$, we uniformly extracted a random number k between 1 and $|\sigma| - 1$, and we considered the prefix σ_{run}, obtained considering the first k events of σ.

Denoted with $\mathcal{A}$ as the activities defined in the reintegration process, the evaluation follows the steps below for each trace σ_{run}:

1) We generate one recommendation, namely an activity $a_{\sigma_{run}} \in \mathcal{A}$.
2) We compute the largest set of traces $L_{\sigma_{run}} = \{\sigma_1, \ldots, \sigma_m\} \subset L_{Similarity}$ such that, for each σ_i, there is a prefix of σ_i that is *similar* to σ_{run}.
3) We compute the largest set of traces $L^G_{\sigma_{run}} = \{\sigma^G_1, \ldots, \sigma^G_m\} \subset L_{\sigma_{run}}$ such that, for each σ^G_i, there is a $\sigma^P_i \in prefix(\sigma^G_i)$ that is *similar* to $\sigma_{run} \oplus \langle (a_{\sigma_{run}}, \emptyset) \rangle$, namely when the recommendation $a_{\sigma_{run}}$ was followed.[6] This also defines the set $L^B_{\sigma_{run}} = L_{\sigma_{run}} \setminus L^G_{\sigma_{run}}$ of traces similar to σ_{run} when the recommendation was not followed.
4) We compute the average KPI for the traces in $L^G_{\sigma_{run}}$: $avgKPIRec(\sigma_{run}) = avg_{\sigma \in L^G_{\sigma_{run}}} \mathcal{T}(\sigma)$.
5) We compute the average KPI for the traces in $L^B_{\sigma_{run}}$: $avgKPINoRec(\sigma_{run}) = avg_{\sigma \in L^B_{\sigma_{run}}} \mathcal{T}(\sigma)$.

Note that $avgKPIRec(\sigma_{run})$ and $avgKPINoRec(\sigma_{run})$ are the average when the recommendation is and is not followed, respectively. The average KPI value of the traces in $L^G_{\sigma_{run}}$ and $L^B_{\sigma_{run}}$ aims to simulate the typical scenarios of similar executions with and without recommendation. Clearly, the definitive assessment would be to test the recommender system

[6]Symbol $\emptyset$ denotes the function with empty domain.

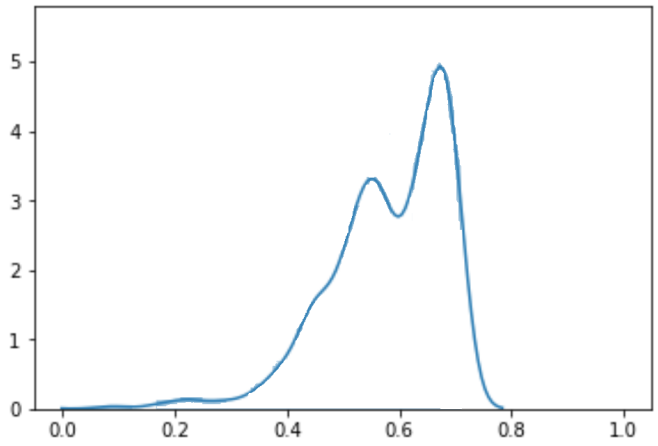


Fig. 3: Distribution of KPI values when recommendations were not followed (Average 0.63). The y axis represents the number of cases.

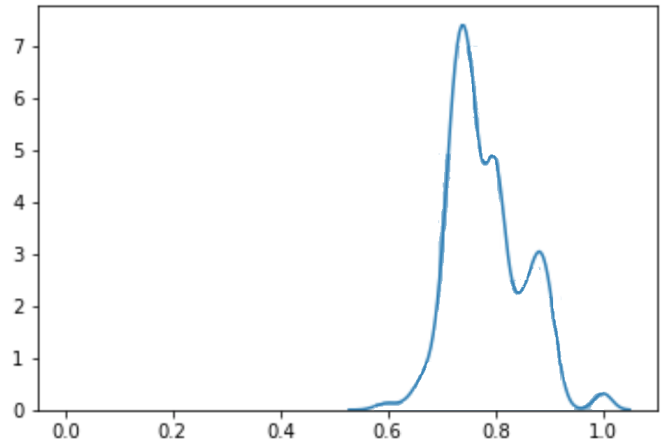


Fig. 4: Distribution of KPI values when recommendations were followed (Average 0.74). The y axis represents the number of cases.

	Traces	**Average Followed**	**Average Not Followed**	Δ
Improvement	1205	0.78	0.58	0.20
Worsening	137	0.56	0.64	-0.08

TABLE II: Number of process instances for which the recommendations led to a statistically significant improvement or worsening of KPI values.

in a real production environment, but the company did not allow this because of the potential consequences on their business.

The above procedure requires one to find the similar traces in $L_{Similarity}$ to a given trace σ_{run}. To achieve this, the prefixes of each trace $L_{Similarity}$ are divided in buckets. A first bucket contains the prefixes with duration of up to one month, where the duration is the difference between the timestamp of the last event and that of the first event of the prefix. A second bucket contains the prefix with duration between one and two months, the third bucket between two and three months, and so on. The prefixes of each bucket are encoded as vectors, analogously to the procedure that was used to learn the prediction oracle (cf. point "Encode Prefix for Training" in Section II-C). Then, a set of clusters is created for each bucket, using Agglomerative Hierarchical clustering along with a principal component analysis to reduce the number of elements of the vectors [10]. The Silhouette score was used to ensure a good quality, while each cluster contained at least 30 traces (ca. 1% of the traces in $L_{Similarity}$). The use of hierarchical clustering has made it possible to set the 30-trace constraint, which motivates the choice of the clustering algorithm itself. Then, we determine the bucket b to which σ_{run} would belong. Finally we encode σ_{run} as vector ν, which ultimately is associated with the b's cluster whose centroid is closer to ν. This centroid contains the prefixes of the traces that are similar to σ_{run}.

It is worthwhile noting that the use of clustering is not part of our prescriptive-analytics framework. It is only used in the case study to determine the traces in a certain log that are similar to a given one.

In sum, given the log $L_{Testing} = \{\sigma^1, \ldots, \sigma^m\}$, we simulated a log of running cases $L_{run} = \{\sigma^1_{run}, \ldots, \sigma^m_{run}\}$. where σ^i_{run} contains the first k events, with k randomly chosen between 1 and $|\sigma^i|-1$. Then, we compute $avgKPIRec(\sigma^i_{run})$ and $avgKPINoRec(\sigma^i_{run})$, for all $1 \leq i \leq m$.

Section IV-D reports on the results obtained through the experimental setup discussed here.

D. Evaluation on Prescriptive Analytics: The Results

Fig. 3 shows the distribution of $avgKPINoRec(\sigma^i_{run})$ for all $\sigma^i_{run} \in L_{run}$. This corresponds to the distribution of KPI values when recommendation is not followed. Recall that, in our case study, a KPI with the value of 1 is the highest value, while 0 is the lowest. Fig. 4 shows the distribution of $avgKPIRec(\sigma^i_{run})$ for all $\sigma^i_{run} \in L_{run}$, namely when recommendations are followed. By comparing Fig. 3 and Fig. 4, one can easily observe how the distribution is certainly closer to 1, when recommendations are indeed followed. The distribution's average indeed increased from 0.63 to 0.74.

The analysis of the distribution of KPI values is certainly valid to gain a general insight into the quality of recommendation. However, this does not directly say how many traces are observing a statistically significant improvement of KPI values when recommendations are followed.

Recall that, for each running trace σ^i_{run}, $L^G_{\sigma^i_{run}}$ and $L^B_{\sigma^i_{run}}$ are the traces similar to σ^i_{run} when the recommendation is or is not followed, respectively. While Fig. 3 and Fig. 4 give a qualitative indication of the presence of a KPI improvement when recommendations are followed, a definitive confirmation requires one to count the number of traces σ^i_{run} for which the average KPI value of traces in $L^G_{\sigma^i_{run}}$ is significantly higher or lower than those in $L^B_{\sigma^i_{run}}$, from a statistical perspective. For each trace σ^i_{run}, we conducted a Z Test to compare the average KPI value of the traces in $L^G_{\sigma^i_{run}}$ with the average of the traces

in $L^B_{\sigma^i_{run}}$. The results are shown in Table II, and illustrates that, when recommendations are followed, there is a significant improvement of KPI values for 1205 out of 3074 traces (ca. 39%), while recommendations had a negative KPI impact on 137 traces (ca. 4%). Columns *Average Followed* and *Average Not Followed* illustrate the KPI values when recommendations were followed or not. For those recommendations with a significant improvement, the improvement was ca. 0.2 when the recommendations were followed. Conversely, for those with a significant worsening of KPI values, the negative effective was more limited: around 0.08. This ultimately means that, when executions followed the recommendations, *the KPI value significantly increased for 39% of the executions*, and the recommendations were counter-productive for few executions, just a limited 4%.

The cases with a significant KPI improvement received recommendations at different moments in time, as shown in Fig. 5. The x axis of the figure represents the time in months, and the y axis indicates the amount of KPI improvement. A point (x, y) of the graph indicates that the traces that received recommendation after x months had an average KPI improvement of y. The curve is obtained via interpolation of discrete points. Fig. 5 highlights that almost every process execution with a significant KPI improvement received a recommendation not later than four months since the execution started. Note that, if one considers every trace of the initial event log of 12,296 traces (cf. Section IV-A), the average case duration is around 12 months with standard deviation of 10.66 months. The average process-instance duration is hence significantly higher than the four months shown in Fig. 5. This shows that *recommendations are almost only useful when followed at the beginning of the case*. From a business viewpoint, this means that, if a customer is at high risk to not find a job, any supporting intervention (i.e. recommendation) should be put in place as soon as possible to have higher chance of success.

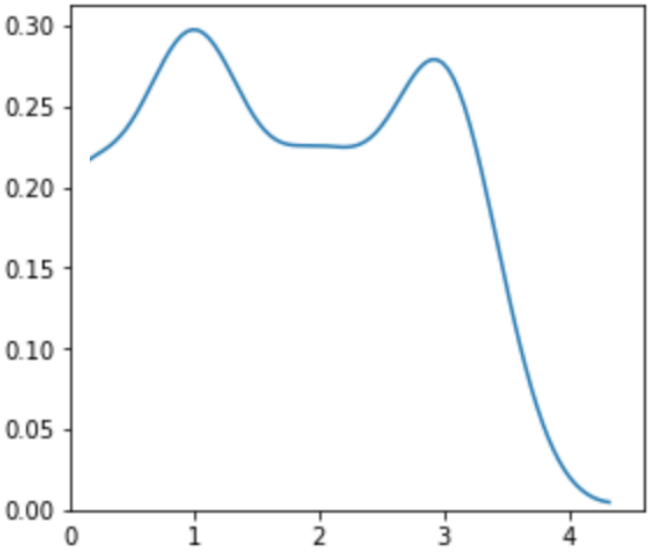


Fig. 5: The relation between the earliness of the recommendation and the amount of KPI improvement when it is followed. The x and y axes respectively represent the time in months when the recommendation was given and followed and the amount of improvement.

V. Related Works

The realm of Process Mining has proposed several research works on Predictive Analytics [3], [12], [13]. Conversely, *"little attention has been given to providing recommendation"* and *"there is still work to be done in this direction [i.e. process-aware recommender systems]"* [12].

In [14], Conforti et al. discusses a recommender system that prioritizes the activities that are offered for executions, without reasoning on additional activities that can be beneficial to put risky process instances back into the right track. To avoid violations of constraints, Maggi et al. [15] aim to provide personalized recommendations to process participants on which activity to work on as next among those in queue for execution. Rozinat et al. propose a framework based on simulation, but they do not provide a concrete operationalization [16]. The same lack of concrete operationalization is also observed in a number of architectures for Process-aware Recommender systems (a.k.a. Operational Support), such as that by Maggi and Westergaard [17]: the infrastructure is only intended to allow researchers to build and deploy their own predictive- and/or prescriptive-analytics algorithms.

Schonenberg et al. [18] and Schobel and Reichert [19] are among the few approaches that propose concrete implementations of prescriptive-analytics modules for recommender systems. However, both approaches base their recommendation on the KPI's average in similar traces. These approaches are certainly valuable when there are sufficiently many traces that are largely the same. However, first, they only focus on the activity information attached to the events, instead of the full payload, as our approach does. Second, they cannot generalize predictive patterns (e.g., on traces that are not very similar), which conversely our framework based on machine-learning is able to do. An empirical evidence of this is in [20]: this paper illustrates that machine-learning techniques are able to provide more accurate predictions (and hence recommendations) with respect to just considering averages.

A very recent proposal of prescriptive analytics based on machine learning is by Weinzierl et al. [21]. However, this proposal is only able to recommend for reducing the remaining time, while we aim at a generic, user-customizable KPI. The prescriptive analytics proposed in this paper can be potentially integrated with that by Teinemaa et al. [22], which helps better prioritize the process cases that require attention, and recommendations.

VI. Conclusions

Process-aware Recommender systems are a new breed of Information Systems that aim to monitor process executions, predict their final outcome and, when the latter is unsatisfactory, suggests corrective actions, in the form of recommen-

dations about the activities to be executed next, to try to get them back on track. While a large share of research has been focused on monitoring and predicting, the aspects related to recommending have been insufficiently considered. It seems that process stakeholders have been assumed to be able to recover from those executions on the basis of their subjective judgment. A few experience reports illustrated that this is not the case: recommendations need to be based on evidence from historical data.

This paper puts forward a prescriptive-analytics framework that relies on process event data. It relies on machine-learning techniques to be able to *(i)* discover the execution patterns, *(ii)* generalize them, and *(iii)* correlate them with KPI values.

By combining the prescriptive analytics with monitoring and predictive analytics, we have been able to design a fully-fledged Process-aware Recommender system.

The framework has been implemented in Python, and evaluated with real-life event data from a Dutch reintegration company. The results illustrate that ca. 39% of the analyzed process executions would have a statistically significant improvement of the KPI values if the personalized recommendations were followed. Recommendations showed to be counter-productive for just 4% of the executions.

We acknowledge that the validity of the evaluation is threatened by the fact that the conclusions have been drawn on the basis of the comparison with similar traces. Conversely, a definitive assessment would require to deploy the system in a production environment, and use it to provide support to a certain fraction of the running cases. At the same time, the rest of the cases would continue as before, without the system. This would allow us to compare the KPI values with and without recommendation. Although the on-the-field experimentation was the initial intention of this research work, it showed itself to be a goal that is hard to achieve. Companies are extremely reluctant to carry on this sort of experimentation, because it can hamper their own business. This is also due to the fact that the trust in the recommender system can only be built if recommendations are coupled with human intelligible explanations that motivate them [23].

The considerations above delineate some directions worthwhile exploring: on the one hand, we aim to test our recommender system in a production environment; on the other hand, we want to equip it with explanations of the recommendations.

Last but not least, our prescriptive analytics only focuses on activities, event and trace attributes, and does not consider the aspects related to the time between events, such as the process-instance duration (time between the first and last activity), or the time elapsed since the last performed activity. We plan to incorporate these time-related aspects, which might additional ingredients towards more accurate recommendations.

REFERENCES

[1] M. Dees, M. de Leoni, W. M. P. van der Aalst, and H. A. Reijers, "What if process predictions are not followed by good recommendations?" in *Proceedings of the Industry Forum at BPM 2019*, ser. CEUR Workshop Proceedings, vol. 2428. CEUR-WS.org, 2019, pp. 61–72.

[2] A. Senderovich, C. D. Francescomarino, C. Ghidini, K. Jorbina, and F. M. Maggi, "Intra and inter-case features in predictive process monitoring: A tale of two dimensions," *Lecture Notes in Computer Science Business Process Management*, p. 306–323, 2017.

[3] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 2, Mar. 2019.

[4] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, and F. M. Maggi, "Complex symbolic sequence encodings for predictive monitoring of business processes," in *International Conference on Business Process Management*. Springer, 2015, pp. 297–313.

[5] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists*. OReilly, 2017.

[6] W. M. P. van der Aalst, *Process Mining: Data Science in Action*. Berlin: Springer-Verlag, 2011.

[7] W. M. P. van der Aalst, M. Pesic, and M. Song, "Beyond process mining: From the past to present and future," in *Advanced Information Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 38–52.

[8] R. P. J. C. Bose and W. M. P. van der Aalst, "Trace alignment in process mining: Opportunities for process diagnostics," in *Proceedings of the 8th International Conference on Business Process Management*, ser. BPM'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 227–242.

[9] L. T. W. Reulink, "Koala," GitHub. [Online]. Available: https://github.com/laurensreulink/koala

[10] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[11] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proceedings of 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017)*, 2017, pp. 477–492.

[12] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortés, "Predictive monitoring of business processes: A survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 962–977, 2018.

[13] M. Camargo, M. Dumas, and O. González-Rojas, "Learning accurate lstm models of business processes," in *Business Process Management*, T. Hildebrandt, B. F. van Dongen, M. Röglinger, and J. Mendling, Eds. Cham: Springer International Publishing, 2019, pp. 286–302.

[14] R. Contorti, M. de Leoni, M. La Rosa, W. M. P. van der Aalst, and A. H. M. ter Hostede, "A recommendation system for predicting risks across multiple business process instances," *Decision and Support System*, vol. 69, no. C, p. 1–19, Jan. 2015.

[15] F. M. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini, "Predictive monitoring of business processes," in *Advanced Information Systems Engineering*. Springer, 2014, pp. 457–472.

[16] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge, "Workflow simulation for operational decision support," *Data Knowledge Engineering*, vol. 68, no. 9, p. 834–850, Sep. 2009.

[17] F. M. Maggi and M. Westergaard, "Designing software for operational decision support through coloured petri nets," *Enterprise Information System*, vol. 11, no. 5, p. 576–596, 2017.

[18] H. Schonenberg, B. Weber, B. van Dongen, and W. van der Aalst, "Supporting flexible processes through recommendations based on history," in *Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 51–66.

[19] J. Schobel and M. Reichert, "A predictive approach enabling process execution recommendations," in *Advances in Intelligent Process-Aware Information Systems: Concepts, Methods, and Technologies*, G. Grambow, R. Oberhauser, and M. Reichert, Eds., 2017, pp. 155–170.

[20] M. Polato, A. Sperduti, A. Burattin, and M. de Leoni, "Time and activity sequence prediction of business process instances," *Computing*, vol. 100, no. 9, pp. 1005–1031, 2018.

[21] S. Weinzierl, S. Zilker, M. Stierle, G. Park, and M. Matzner, "From predictive to prescriptive process monitoring: Recommending the next best actions instead of calculating the next most likely events," in *Proceedings of the 15th International Conference on Wirtschaftsinformatik*, 2020.

[22] I. Teinemaa, N. Tax, M. de Leoni, M. Dumas, and F. M. Maggi, "Alarm-based prescriptive process monitoring," in *Proceedings of Business Process Management Forum - BPM Forum 2018*, ser. Lecture Notes in Business Information Processing, vol. 329. Springer, 2018, pp. 91–107.

[23] C. Molnar, *Interpretable Machine Learning*, 2020, available at https://christophm.github.io/interpretable-ml-book/.