A General Framework For Predictive Process Monitoring

A Preprint

Lin Mi

15527504293@163.com

December 3, 2022

Abstract

Predictive process monitoring is a subdomain of process mining and has recently received traction in academia and companies. Predictive process monitoring is a creative process that requires a large number of different skills and knowledge. There is yet to be a general framework to establish a predictive process model with high precision and straightforward interpretation. This paper tried to define a general framework of Predictive Process Monitoring and built an R package: ppmr, according to this framework.

***K*eywords** process mining · predictive process monitoring · predictive model · Interpretability machine learning · data science

1 Introduction

Over the last few years, more and more companies and organizations have been concerned about identifying bottlenecks and the chance of a process to gain better results. Companies and organizations are seeking a way to reduce time, reduce costs, balance resource utilization, minimize risk, improve quality, improve worker well-being and maximize productivity(Santos Garcia et al. 2019). Process mining is an emerging and relatively young research field. Process mining aims to help people to understand the real processes and to capture meaningful findings during the real execution of processes. The main challenge of process mining is to build an appropriate and explicit process model. Furthermore, use tools to diagnose process problems by observing dynamic behavior(Van der Aalst and Weijters 2004). Process mining provides a new possibility for organizations to solve process problems and optimize process efficiency.

Predictive process monitoring is a subdomain of process mining and has recently received traction in academia and companies. Predictive process monitoring (PPM) is a method that takes historical process data - a set of completed business process executions - as input. Then it uses machine learning techniques to predict process indicators such as the remaining time, following activity, or any other outcome of a trace of running process instances (Spree 2020). The principal value of predictive monitoring is to provide information about the real process to take proactive and corrective actions to improve process performance and reduce risks in real-time.

The value of predictive process monitoring is undeniable, and the problem of predictive process monitoring has received substantial attention in the past years. A considerable number of research have been put forward to address the issue of predictive process monitoring. However, researchers use different ways to deal with predictive process monitoring problems, such as other event data processing methods, algorithms, and prediction goals. This makes the predictive process monitoring research field highly complex, and it is hard for ordinary people to apply predictive process monitoring to their work problems. Therefore, The contribution of the paper is to propose a common framework of predictive process monitoring; This can effectively promote the research of proactive process monitoring and help people to correctly apply proactive

Table 1: Event Log About Patiens

handling	patient	employee	time
Registration	1	r1	2017-01-02 11:41:53
Triage and Assessment	1	r2	2017-01-02 12:40:20
Blood test	1	r3	2017-01-05 08:59:04
MRI SCAN	1	r4	2017-01-05 21:37:12
Discuss Results	1	r6	2017-01-07 07:57:49

process monitoring to solve real process prediction problems. Furthermore, this paper has developed an R package -ppmr- for predictive process monitoring.

The article is organized as follows: In Section 2, introduction to the basic knowledge of process mining in a Summary way. Section 3 introduces predictive process monitoring related work, and in Section 4, the common framework of predictive process monitoring is introduced. Section 5 presents the ppmr R packages about this common framework. Finally, a conclusion is given in Section 6, including directions for future research.

2 PROCESS MINING OVERVIEW.

Process mining is considered a component of data science. It combines the method of transforming available data into value and process science. It is a broader discipline combining information technology and management science knowledge to improve and operate the operation process(Dakic et al. 2019). Process mining uses event logs and related information to observe business activities: (1) discover process models; (2)conformance checking and compare the event log with the defined process model; (3) Enhance and expand the model by combining performance, resource details and bottleneck information(Rozinat et al. 2009).

The research field of process mining can be divided into three major parts: (1)Process model discovery, (2)Conformance checking, and (3)Enhancement(Van Der Aalst 2016). In addition, the researchers made a more detailed division, (Santos Garcia et al. 2019)divid process mining research into four This research field can be divided into three major categories (1)Discovery, (2)Conformance, (3)Enhancement, (4)supporting area. Each major research field can be divided into other sub-research filed 1.

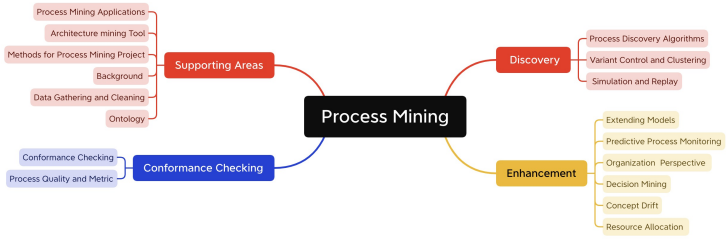

Figure 1: Process Mining Research Filed.

2.1 Event log

In any problem of data mining or data science, data is the basis of all analysis and modeling in the field of process mining. Data and information about organizations or business processes are maintained and recorded in an event log system. All events were recorded from the start to the end(Ahmed, Faizan, and Burney 2019). The start point of process mining is the event log; table 1 illustrates the specific information present in an event log used for process mining.

Each row is an event in the process, and each event in the log needs to refer to a single process instance, often referred to as a case. In table 1, the case id is patient, and the case id of the first five rows is 1. Moreover, events related to some activity represent what happens in the process. In table 1, handling represents activity, and events refer to activities like Registration, X-Ray, Blood test, RI SCAN, and Check-out. In process mining, the case id and activity is the minimum component of the event log, meaning every event

log must include the case id and activity. Moreover, the events in the case need to be sorted. For example, the execution of activity Registration for Case 1 start time is 2017-01-02 11:41:53, and the execution of activity Triage and Assessment for Case 1 start time is 2017-01-02 12:40:20. Without sorting information, it is impossible to find causal dependencies in the process models. Although the minimum component of the event log is case id and activity with the order, 1 also shows other information about the process, such as resource(employee), timestamp(time), lifecycle transition(registration_type), In the context of process mining, these properties are called attributes. These attributes can be used to analyze process performance, identify process bottlenecks, or any other possible analysis.

On occasion, the event log data file contains such fields.

1. Case ID - Action set order of events log.
2. Activity - Executed activities in the event log.
3. Resource - The participant of the event log.
4. Timestamp - The time and date the event log was recorded.

2.2 Process discovery

Process modeling is a graphical representation of a business process or workflow. Just like the flow chart, each step of the process is drawn, so that the tasks in the process can be summarized end-to-end in the context of the business environment. There are various process modeling notations, such as BPMN, Petri Nets, and Transition Systems.

The process modeling behavior provides the visualization of the business process, making it easier to check, so users can understand how the process works in its current state and how to improve it. Other benefits of process modeling include the following:

1. Improve efficiency – process modeling helps improve processes and helps business people improve productivity by saving time.
2. Improve transparency – modeling provides a clear overview of the process, identifying the starting and ending points and all the steps in between.
3. Ensure best practices – use process models to ensure consistency and standardization across the organization.
4. Create understanding – make it easier for users across the organization to communicate with each other by using a standard process language.
5. Business coordination – support the coordination of personnel, systems, and information within the organization to help the business strategy.

Process discovery is the most challenging research field in process mining. Process discovery aims to deliver a process model with quality, and it is expected to be understandable by avoiding unnecessary complexity and providing acceptable accuracy, balancing recall, precision, and generalization(De Weerdt et al. 2012). According to(Van Der Aalst 2016), process discovery can support managers to quickly answer questions, e.g., what happened in the past? What are the highest frequent paths? When and why do organizations and people deviate? How is the frequency distribution according to each processing path? How to redesign a process to improve its performance? How are the process instances?

There are many process discovery algorithms. According to (Santos Garcia et al. 2019), the most popular process discovery algorithm is Heuristic Miner(Van der Aalst et al. 2003). One possible explanation for this result is that these algorithms can handle noise and anomalies in unstructured processes. Other algorithm include:Alpha miner(Van der Aalst and Dongen 2002), Social miner(Van der Aalst and Song 2004), Generic miner(Van Dongen et al. 2005), Integer Linear Programming miner(Jansen-Vullers, Aalst, and Rosemann 2006), Fuzzy miner(Günther and Van Der Aalst 2007), Declarative miner(Lamma et al. 2007), Region-based miner(Bergenthum et al. 2007), Inductive miner(Leemans, Fahland, and Van Der Aalst 2014), Multi-paradigm miner(De Smedt, De Weerdt, and Vanthienen 2014) and Fodina(Broucke and De Weerdt 2017).

2.3 Conformance checking

Process conformance is a process mining type responsible for measuring the quality of process models. Four quality dimensions are usually considered to describe the quality of the process model(Van Der Aalst 2016).

Conformance checks can counter the model against the reality obtained through the event log on the system. According to (Van Der Aalst 2016), conformance checking can be used to verify the accuracy of the documented

process, point out different cases, try to understand their common points, and determine deviations in the process. In addition, conformance checking can be used for audit purposes to calculate the efficiency of discovered process models and improve new or existing models. Conformance checking is used often, making it one of the pillars of process mining.(Alves de Medeiros and Aalst 2008) also established the main "C-level" interests related to process compliance, such as: how are the compliance of execution cases with standardized and documented process models, what are the main problems, how often exceptions occur, and where they violate business rules.

In summary, The mainly conformance checking is used for:

1. Fitness: the ability to replay the event log, high fitness means the process model can replay most traces in the event log.
2. Precision: the ability to avoid behavior unrelated to the used event log in the discovery process, like avoiding underfitting concept;
3. Generalization: the ability to accept new similar events related to previous events used for discovery, like avoiding overfitting concepts.
4. Simplicity: the process model should be as simple as possible.

Conformance checking is one of the technologies for discovery in process mining and is the most necessary and exciting part of process mining evaluation activities (Ahmed, Faizan, and Burney 2019). Mainly conformance checking is used for:

1. Repairing models.
2. Auditors.
3. Emerging the arrangement of business processes.
4. Algorithm discovery, evaluating the process
5. Even lot connecting and process model.

The most common conformance-checking methods include:

1. Token-based play conformance checking.
2. Footprint method for conformance checking.
3. Alignment base conformance checking.

2.4 Enhancement

Enhancement extends or improves existing process models by using case information (such as timestamps and resource information) to analyze throughput time, discover patterns and case frequency, identify bottlenecks and infrequent behaviors, discover feedback loops, and analyze time trends and waiting times(Dakic et al. 2019). After process discovery and conformance checking, extended information can be added to the process model. 2 shows the basic framework of process mining.

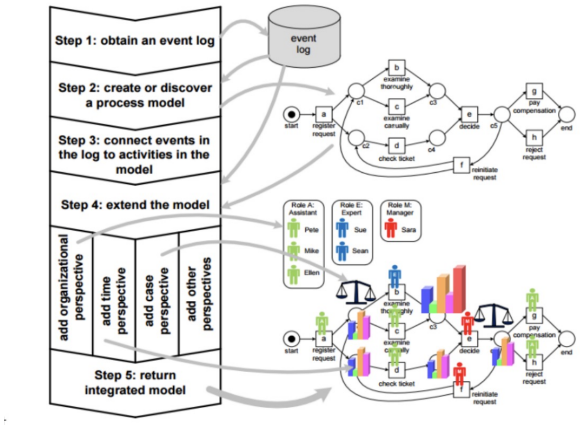

Figure 2: Enhancement For Process Model.

There are many perspective of information that can be added:

1. Organizational perspective.
2. Control flow perspective.
3. Time performance perspective.
4. Any other perspective of the process.

3 Related work

Predictive monitoring of business processes is a challenging topic of process mining, which involves the prediction of process indicators of running process instances. Predictive process monitoring aims to predict the future indicator of the process during a running process execution(Maggi et al. 2014). For example, predicting key performance indicators can help people focus on the long-term value of the business. Predictive process monitoring uses the historical data of the same type of executed processes to predict the results of running processes. The set of historical data consists of events corresponding to the execution of activities of each process instance. There are three types of prediction : (1)prediction related to numeric or continuous measures of interest, such as the remaining execution time/cost of a caseVan der Aalst, Schonenberg, and Song (2011). (2)predictions related to categorical or boolean outcomes, such as delay/riskDi Francescomarino et al. (2016). (3)predictions related to sequences of future activitiesBernard and Andritsos (2019). There are two main approaches to build the predictive model: (1) methods relying on the explicit process model, e.g., annotated transition systems(Van der Aalst, Schonenberg, and Song 2011). (2) approaches relying on a statistic or machine learning methods. This way, predictive process models build by encoding event logs as input data.

3.1 Time prediction

There's a lot of research on time prediction. (Van der Aalst, Schonenberg, and Song 2011) puts forward a method of a transition system. Specifically, the information on the elapsed time, dwell time, and remaining time of each state of the transition system is reported. This information is then used to predict the completion time of the ongoing tracking. (Polato et al. 2014, 2018) did further research and used machine learning models to annotate transition systems. In (Folino, Guarascio, and Pontieri 2012, 2013), The annotated transition system is combined with the context-driven predictive clustering method. (Ceci et al. 2014) propose sequence trees models to predict the completion time and the next future activity. (Rogge-Solti and Weske 2013)use generally distributed transitions stochastic Petri nets (GDT-SPN) to predict the remaining execution time. In (Rogge-Solti and Weske 2015), the author uses the time since the last event to make a more accurate prediction of the remaining time and estimate the probability of missing the deadline. In (Pandey, Nepal, and Chen 2011), the Hidden Markov model (HMM) predicts the remaining time. The above methods all depend on the process model, (Dongen, Crooy, and Aalst 2008) using nonparametric regression and activity duration and incidence, as well as other case-related data, to develop a method to predict the remaining cycle time of cases. (Senderovich et al. 2017)Propose and evaluate different strategies for extracting features between cases to predict completion time.

3.2 Categorical Predictions

This kind of prediction relies almost on the implicit model. (Maggi et al. 2014) propose a framework for predicting a predicate's fulfillment (or the violation) in an ongoing execution. In(Leoni, Van der Aalst, and Dees 2014; De Leoni, Aalst, and Dees 2016), a framework is proposed to correlate different process characteristics for more detailed analysis and prediction. (Tax et al. 2017; Navarin et al. 2017) use long short-term memory (LSTM) neural networks to predict the next activity and remaining time. (Di Francescomarino et al. 2016) introducing a clustering preprocessing step for predictive process monitoring; in this approach, first cluster event log into different groups, then build a classification model by each event log group. (Teinemaa et al. 2016) applied text mining methods into predictive process monitoring and implemented different combinations of text mining and classification. In(Márquez-Chamorro et al. 2017), an approach based on an evolutionary algorithm is proposed.

3.3 Activity Sequence Predictions

Predicting business process behavior is important. (Evermann, Rehse, and Fettke 2017, 2016) uses deep learning with recurrent neural networks to predict the next event in the business process. (Tax et al. 2017)

use LSTM with activity and timestamp-based coding to provide predictions of the next activity and its timestamp.

4 Framework of predictive process monitoring

So far, many researchers have used different steps and methods to solve the PPM problem. The steps and data processing methods used by these researchers to solve the PPM problem are different. This leads to the complexity of PPM research, and it is difficult for companies or institutions to choose appropriate PPM methods to solve their problems in data science.

Predictive process monitoring is a creative process that requires a large number of different skills and knowledge. At present, there is no standard framework to execute the Predictive process monitoring project. This means that the success or failure of the Predictive process monitoring project depends to a large extent on the specific personnel or team executing the project, and successful practices may not be repeated in the entire enterprise. Predictive process monitoring needs a standard method, which will help transform business problems into data mining tasks, suggest appropriate data conversion and data mining technologies, and provide methods to evaluate the effectiveness of results and record experience.

To solve this problem, this paper tries to define a process model, which provides a framework for executing the Predictive process monitoring project. The framework is independent of the industry department and the technology used The framework is designed to make the predictive process monitoring project more reliable, repeatable, easier to manage, and faster, make the prediction of the predictive model more accurate, make the prediction more timely, and generate some insights on data business through the Predictive models.

The framework is divided into two stages: the training stage and the prediction stage. In the training phase, one or more prediction models are constructed using the information contained in the execution log 3. In the prediction phase (runtime or prediction phase), (one or more) prediction models are utilized to obtain predictions related to one or more ongoing execution tracks 4.

The training process includes:

1. Define the predictor of the event log.
2. Enrich the event log.
3. Prefix event log.
4. Encoding prefix event log.
5. Bucketing prefixes.
6. Train predictive model.
7. Interpret predictive Models.

These steps are not entirely sequential; for example, steps 4 and 5 are not absolute, and the order of the two can be converted. Whether to convert depends on the purpose of the analysis and the method used. The prediction phase includes:

1. Prefix event log.
2. Encoding prefix event log.
3. Bucketing prefixes.
4. Apply predictive model

In the prediction stage, the order of steps 2 and step 3 is also not fixed, and the specific order depends on the order of steps 4 and steps 5 in the training stage.

A general predictive process monitoring framework is beneficial. For practitioners, the standard framework provides guidance, helps build projects, and advises on each task in the process. In addition. Another function of the standard framework is to communicate and record results, thus forming an efficient and repeatable process. A general framework is also conducive to research, and it is also beneficial for researchers to compare the results of various methods and conduct further research.

4.1 Define predicate of event log

The first step in building a prediction model is determining the prediction goal. In the event log, each process instance has a trace, a series of events. These events have different characteristics. The necessary

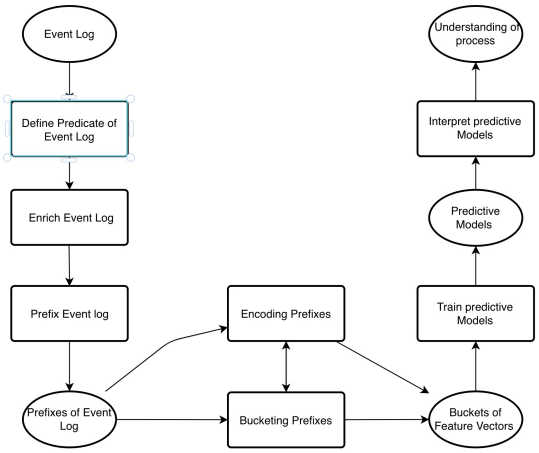

Figure 3: Training Phase.

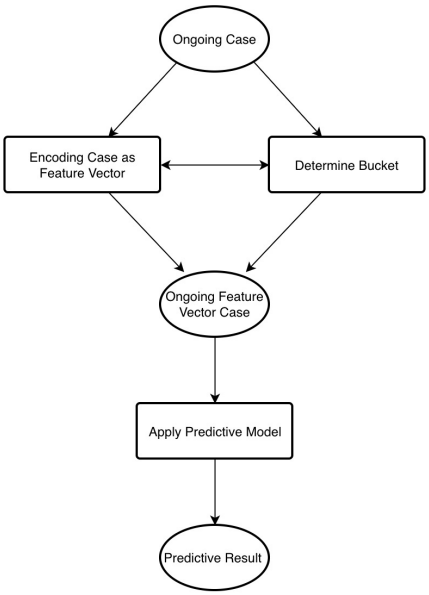

Figure 4: Predicting Phase.

characteristics are case id and activity. Other standard characteristics include resource, activity lifecycle, cost, etc. Other features related to the process instance can also be added, such as the gender of the patient. The value of the prediction target can be the value of a feature in the event log. For example, to predict the next activity after the fifth activity in the trace, you need to generate the sixth activity of each case in the event log as the target variable. However, in most cases, the value of the target variable needs to be calculated from the characteristics in the event log, such as the case duration or risk. How to define and calculate this prediction target depends on the problems encountered and the data used. In (Zandkarimi et al. 2020), the 'Define analysis use case' step is defined. In this step, you need to select a dependent feature, select an independent feature, and determine how to filter the event log. The concept of choosing a dependency feature here is similar to that of defining prediction, and it is used to determine the analysis or prediction target. In the framework mentioned in this article, all available features in the event log can be retained. If necessary, the features can be filtered in the modeling step. In addition, since the event log will be divided into buckets in the next step, the event log can be filtered.

At this stage, you can also perform a fundamental analysis of the event log to understand the information contained in the event log. The angles of analysis include:

1. Organizational perspective: for example, analyze who executes the work, who specializes in a certain task, whether is there a risk of brain drain, who transfers work to whom, the resource frequency, each resource, or resource-activity combination in what percentage of cases it is present.
2. Control flow perspective: Analyze the trace length and its distribution, analyze the Start and End points of the trace, analyze the distribution of activity, Analyze rework in case, the trace coverage, repetition of the activity
3. Time perspective: Analyze Throughput time, Processing time, and idle time.

Here are three common analysis angles. Through preliminary analysis, we can have an understanding of the overall situation of the event log, and these analysis results pave the way for our second step.

4.2 Enrich event log.

In the event log, the necessary features are case id, activity, and event. Other standard features include resource, activity lifecycle, cost, etc. Other features related to process instances can also be added to the event log. In addition, more new features can be generated based on existing features. In(Zandkarimi et al. 2020), five rich event log perspectives are defined, including (1) Control flow perspective, (2) Data flow perspective, (3) Resource perspective, (4) Time perspective, and (5) Conformance perspective. The decadent event log method mentioned in this article is very enlightening, but it is only partially applicable to the framework mentioned in this article. In this article's Encoding prefix event log step, different methods will be selected to process the control flow information in the event log. For example, some encoding methods can ultimately retain the control flow information in the event log. Therefore, this article partially refers to the rich event log method in (Zandkarimi et al. 2020).

1. consider the view of data flow. Here, we focus on the data attributes attached to the event. These attributes may refer to the event itself or the characteristics of the case to which the event belongs. The methods of enriching event logs include Average/Maximum/Minimum/Sum/count(any other aggregation function) of the value of Characteristic Until the Current Event.
2. consider the resource perspective. The resource and organization perspectives focus on the resources of the execution event or corresponding activity instance. Resources can be grouped in different organizational entities (groups, roles, positions, departments, etc.). Features that can be derived include:(1)frequency of resources at case levels Until the Current; Event.(2)involvement of resources at case levels Until the Current Event.
3. The time perspective refers to various durations in the process, two features are added to 2 considered here:(1)throughput times at case levels Until the Current Event.(2)involvement of resources at case levels Until the Current Event.

Here are three feature derivation angles and some specific derived features. Of course, other derived features can also be considered.

4.3 Prefix event log.

In the prediction process monitoring, our goal is to predict the tracking of incomplete cases (stored in the event flow), not the tracking of completed cases (stored in the event log). To capture such partial tracking, we define a function that returns the first k events of the (completed) situation tracking.

Given the event log of a complete case of a business process and the prefix case of the process obtained from the event flow, we hope to predict the performance measurement of this prefix case in the future. For example, we may want to predict the completion time (or remaining time) of the case or the results at completion. Prediction is based on the predictor's understanding of the history of the process up to the prediction point and the future up to the prediction point. In real life business processes, the number of uncertainties increases over time, so forecasting tasks become more difficult and often less accurate. Therefore, the prediction is usually at a specific point in the future, that is, time range h. The selection of H depends on the development speed of the process and the prediction target.

Using all possible prefixes can cause multiple problems. First, compared with the number of tracks, a large number of prefixes will greatly slow down the training speed of the prediction model. Secondly, if the original case length is very heterogeneous, the longer trace will produce more prefixes than the shorter trace. Therefore, the prediction model is biased towards the longer case. Therefore, only prefixes of up to a certain number of events are usually considered. For example, only prefixes with up to 20 events are used.

Considering that at runtime, we need to predict partial tracking rather than complete tracking, this choice is very natural. The prefix log is used for training to ensure the comparability between our training data and test data. Therefore, in this step, we need to determine the length of the prefix and convert the event log to the prefix log.

4.4 Encoding prefix event log.

All prefix tracking needs to be represented as a fixed-length feature vector to train the classifier. This can be achieved by applying trace abstraction techniques, such as considering only the last event traced. Choosing the right abstraction is complex, requiring a balance between generality and information loss.

4.4.1 static

In static encoding, the classification variables are hot-coded. The encoding of case attributes is quite simple. Because they remain the same in the whole case, they can be added to the feature vector "as is" without losing any information.

4.4.2 last state/last m state

In this encoding method, only the last available data snapshot is used. Therefore, the size of the feature vector is proportional to the number of event attributes and is fixed throughout the execution of the case. One disadvantage of this method is that it ignores all information that has occurred in the past and only uses the latest data snapshot. This coding can be easily extended to the last m state to alleviate this problem. In this case, the size of the feature vector increases by m times.

4.4.3 Aggregation

The final state coding has obvious disadvantages regarding information loss, ignoring all data collected in the early stage of tracking. Another method is to consider all events since the beginning of the case but overlook the sequence of events. This abstract approach paves the way for several aggregate functions that can be applied to the values of event attributes throughout the case. In contrast to the last state code, the aggregation code considers all events in the case prefix, not only the last m events. Depending on the attribute's data type, each attribute is encoded as one or more characteristics using different aggregate functions. Suppose an attribute is of numerical type (for example, sales, cost). We can map this attribute to a feature through a numerical aggregation function, such as sum, average, minimum or maximum, variance, median, etc. However, if the attribute is classified (for example, risk level, activity), we map the attribute to a characteristic of each value in the attribute field by applying the "count" aggregation function. For example, the possible value of attribute risk level is {A, B, C, D, E}. We map this attribute to five characteristics {Risk Level - A, Risk Level - B, Risk Level - C, Risk Level - D, Risk Level - E}. For a given prefix of a case, the risk level - A is the number of times that the attribute risk level has a value equal to "a" in the events in

the prefix, as well as for the characteristic risk level - B, risk level - C, etc. In other words, the resulting feature corresponds to the absolute frequency of each possible value in the attribute domain within the case's current prefix.

In a word, classified variables, such as activities/resources, can use counting to aggregate, and numerical variables, such as cost, can use numerical aggregation functions to process.

4.4.4 index

Although aggregation coding utilizes information from all executed events, it still shows information loss by ignoring the sequence of events. The idea of index-based coding is to use all possible information (including order) in the trace, and each event attribute of each executed event (each index) generates a feature. In this way, the lossless coding of the trace is realized, which means it is possible to recover the original trace according to its feature vector completely. One disadvantage of index-based coding is that the length of feature vectors increases with each event executed

4.4.5 tensor encoding

Tensors are vectors and matrices that promote potentially higher dimensions. Unlike traditional machine learning algorithms, tensor-based models do not need to use two-dimensional $n \times m$ The form m encodes the input, where n is the number of training instances and m is the number of features. Tensor encoding is required when building a predictive model using the deep learning model. Instead, prefix trace can take shape $n \times t \times m$ The three-dimensional tensor of p is used as input, where t is the number of events and p is the number of event attributes or features derived from each event. In other words, each prefix is represented as a matrix, where rows correspond to events and columns correspond to the characteristics of a given event.

The encoding methods mentioned here can be combined. Static encoding can extract different data types (case attributes) from tracking and combine static encoding with one of the other three kinds of encoding. In addition, the last m state can be combined with Aggregation encoding.

4.5 Bucketing prefixes.

Some predictive process monitoring methods based on machine learning train a single predictor on the whole event log. Most existing predictive process monitoring methods train multiple classifiers instead of one classifier. In particular, prefix tracking in the history log is divided into several buckets, and different classifiers are trained for each bucket. At runtime, determine the bucket most suitable for the ongoing situation, and apply the corresponding classifier to predict.

1. zero buckets: All prefix traces are considered in the same bucket. Therefore, a single predictor is suitable for all prefixes in the prefix log(De Leoni, Aalst, and Dees 2016).
2. KNN: In this bucket method, the offline training phase is skipped, and buckets are determined at runtime. For each running prefix tracking, k nearest neighbors are selected from the history prefix tracking, and the classifier is trained based on these k neighbors (at run time). This means that the number of buckets (and classifiers) is not fixed but grows with each event executed at runtime. (Maggi et al. 2014) use the string editing distance on the control flow to calculate the similarity between prefix traces. All instances that exceed the specified similarity threshold are considered neighbors of the running trace. If the number of neighbors found is less than 30, the first 30 similar neighbors are selected regardless of the similarity threshold.
3. Prefix length bucketing: Each bucket contains a prefix of a specific size. For example, the nth bucket includes a prefix that has executed at least n events. Build a classifier for each possible prefix length(Velmurugan et al. 2021).
4. cluster bucketing: Each bucket represents a cluster, which is generated by applying the clustering algorithm to the encoding prefix. Train a classifier for each generated cluster, and only consider the historical prefix belonging to the specific cluster. At runtime, determine the cluster of the running case based on its similarity to each existing group, and apply the corresponding classification(Leontjeva et al. 2016).
5. Domain Knowledge: You can define barrel functions based on manually built rules. In this approach, input from domain experts is required. For example, the generated bucket can refer to the context category(Ghattas, Soffer, and Peleg 2014).

4.6 Train predictive model.

Under the framework of this article, any regression or classification model can be used to build a predictive model. The existing prediction process monitoring methods have tried different classification algorithms. The most popular choice is the decision tree, which has obvious advantages in the interpretation of results (Di Francescomarino et al. 2016). Another popular method is random forest. This model usually achieves better prediction accuracy than a single decision tree, but it is more difficult to explain (Verenich et al. 2016). Other machine learning models, support vector machines (SVM) and generalized step-up regression models are also used in predictive process monitoring (Leontjeva et al. 2016), but their performance is not as good as that of random forests. Gradient lifting trees and xgboost Elkhawaga, Abu-Elkheir, and Reichert (2022) show very good results, which are generally better than random forests. In recent years, deep learning has been used in the field of predictive process monitoring. Generally, deep learning technology can provide results with very high accuracyEvermann, Rehse, and Fettke (2017). despite the emerging attention towards deep learning also in predictive process mining, stacking feature construction and shallow machine learning algorithms can still outperform various process predictor competitors included deep learning ones(Appice, Di Mauro, and Malerba 2019).

4.7 Interpret predictive Models.

Machine learning models are widely used in classification or regression tasks. Due to the improvement in computing power, and the availability of new data sources and methods, ML models are becoming more and more complex. The model created by boosting, bagging neural networks, and other technologies is a black box. It isn't easy to understand the relationship between input variables and model results. They are used because of their high performance, but the lack of interpretability is one of their weakest aspects.

In many scenarios, we need to know, understand or prove how the input variables are used in the model and how they affect the final model predictions.

For the prediction model, we expect to meet three requirements:

1. For each prediction of the model, people should be able to understand which variables affect the prediction and the degree of impact. The final predicted variable attribution.
2. For each prediction of the model, people should be able to understand how the model prediction will change if the input variables change. Assumptions.
3. Verification of predictions For each prediction of the model, one should be able to verify the strength of the evidence that supports that particular prediction.

There are two ways to meet these requirements. One is to use only models that meet these conditions through design—white box models, such as linear regression or decision trees. In many cases, transparency comes at the expense of performance. Another approach is to use an approximation interpreter - a technique that can only find approximate answers but applies to any black box model. Therefore, we can try to understand the model from three perspectives:

1. Understand the importance of different variables
2. How different variables affect predictions
3. How changes in variables affect predictions

4.7.1 Importance of variables

Understanding the importance of the variables used in the model is essential. By analyzing the importance of variables in the model, we can remove variables that do not affect the model's prediction to simplify the model. In addition, identifying the essential variables may lead to discovering new factors involved in specific mechanisms. The methods for evaluating the importance of variables can generally be divided into two groups: model-specific and model-agnostic. For linear models and many other models, there are ways to use specific elements of the model structure to assess the importance of explanatory variables. These are model-particular methods. For a linear model, the value of the normalized regression coefficient or it is corresponding p value can be used as the variable importance measure (Grömping 2007).

For tree-based collections, such measurements can be based on using specific variables in a particular tree. A good example is the variable importance measures of a random forest model based on out-of-pocket data (Archer and Kimes 2008). Some algorithms for calculating the importance of variables are based on the

variation of the random forest (Kursa, Jankowski, and Rudnicki 2010). These methods are usually based on a specific algorithm. The model agnostic approach does not make any assumptions about the model structure. Therefore, it can be applied to any prediction model or model set. In addition, it allows a comparison of the importance of explanatory variables between models with different structures. (Molnar et al. 2020) propose a new sampling mechanism for the conditional distribution based on permutations independent subgroups.(Fisher, Rudin, and Dominici 2019) offer model class reliance (MCR) as the range of VI values across all well-performing model in a prespecified class. And Variable importance (VI) tools describe how much covariates contribute to a prediction model's accuracy. The main idea is to measure how much the model's performance will change if the influence of selected explanatory variables or a group of variables is deleted. If a variable is necessary, we expect that the model's performance will deteriorate after arranging the values of the variables—the greater the performance change, the more influential the variable.

4.7.2 How variables affect predictions

When trying to understand the model's prediction of a single observation, the most common question may be: Which variables contribute the most to this result? There is no single best way to answer this question. (Staniak and Biecek 2018) use Break down Plots for Additive Attributes to answer this question. The basic idea is the prediction $f(x)$is an approximation of the expected value of the dependent variable Y given values of explanatory variables $\underline{x}$. The underlying idea of BD plots is to capture the contribution of an explanatory variable to the model's prediction by computing the shift in the expected value of Y while fixing the values of other variables. The Break down Plots for the Additive Attributes method cannot have an interactive model, and its results depend on the order of the explanatory variables used in the calculation. One solution to this problem is to find a sequence and place the most important variables at the beginning. (Staniak and Biecek 2018) proposed to Break down Plots for Interactions to analyze models with interactions. In addition, (Lundberg and Lee 2017) proposes another method to solve the sorting problem. It is based on the idea of averaging all (or a large number of) variable attribute values that may be sorted

4.7.3 How changes in variables affect predictions

The Ceteris paribus Profiles method is based on the principle of other things held constant. The model prediction changes caused by variable value changes are used to evaluate the impact of the selected explanatory variables. This method checks the influence of explanatory variables by assuming that the values of all other variables are unchanged. The main objective is to understand how variable values change model predictions(Biecek 2018). Ceteris paribus Profiles show the dependence of conditional expectations of dependent variables (responses) on the values of specific explanatory variables. If the model has many explanatory variables, the Ceteris paribus Profiles method may not be appropriate because we may eventually get a large number of graphs. In this case, selecting the most interesting or important configuration file may be helpful. To give Ceteris paribus Profiles importance, we can use the concept of profile ossifications. The greater the influence of explanatory variables on predicting a specific case, the greater the fluctuation of the corresponding Ceteris paribus Profiles. The curve will be flat or almost unchanged for variables that have little or no impact on the model predictions. In other words, the value of Ceteris paribus Profiles should be close to the model's predicted value for a specific instance. Therefore, the sum of the differences between the profile and the predicted values (covering all possible explanatory variables) should be close to zero(Kozak 2020; Kozak and Biecek, n.d.).

Understanding the model is very important. Understanding the model can help us understand the relationship contained in the data and identify the critical business behind the data. Therefore, after building the model, understanding the model is a crucial step for proactive process monitoring. For example, we need to predict the remaining time of the process. Suppose we find that the presence or absence of activity is significant for the final prediction, which prompts us to think that there may be a bottleneck or other reasons in the real process.

5 ppmr:A r packages for predictive process monitoring

ProM is the most commonly used tool for process mining. It provisions an extensive range of process mining methods and also provided the extensible framework form of plug-ins(Van Dongen et al. 2005). Other tools include Disco, Enterprise suite concept, Mosaic, etc. R and Python are the two most commonly used tools in the field of data science(Larose and Larose 2019), R language is an open source statistical calculation and drawing language, which is free of charge, easy to learn, and has rich expansion packages. Python is a programming language, widely used in Web applications, software development, data science and

Table 2: Enrich,Prefix Event Log and Define Predicate

patient	handling	employee	registration_type	time	predicate
1	Registration	r1	start	2017-01-02 11:41:53	Discuss Results
1	Triage and Assessment	r2	start	2017-01-02 12:40:20	Discuss Results
1	Registration	r1	complete	2017-01-02 12:40:20	Discuss Results
1	Triage and Assessment	r2	complete	2017-01-02 22:32:25	Discuss Results
1	Blood test	r3	start	2017-01-05 08:59:04	Discuss Results

machine learning (ML). It also has a concise syntax and an easy-to-use tool library. But at present, neither R nor Python supports process mining very well. In Python, there is only one library related to process mining, i.e. PM4Py(Berti, Van Zelst, and Aalst 2019). In R, the R package related to process mining is bupaR(Janssenswillen et al. 2019). However, neither R nor Python provides tools to solve predictive process monitoring problems. The R package proposed in this paper - ppmr -attempt to fill this gap. This package uses the framework mentioned in this article to provide R users with a tool to solve predictive process mining.

5.0.1 Enrich,prefix event log and define predicate

Defining a predicate depends on the problems encountered in the specific business. In the ppmr package, two standard predicates are defined: the duration of the next activity and process. In addition, only time and resource perspectives are currently considered for feature derivation in the ppmr package.

In the ppmr package, use the `enrichEventlog` function to Enrich, prefix event log, and define predict. This function has three primary parameters:

1. eventLog : A event log
2. prefix_num : prefix length of trace
3. mode : predicate-activity or duration

The following is an example of this function. The data used is 1. In this example, the defined prediction target is the next activity.

```
library(bupaR)
library(ppmr)
eventdata <- enrichEventlog(eventLog = patients,prefix_num = 4,mode = "activity")
```

5.0.2 Bucketing prefixes

You can skip this step if you use all the data in the event log to build the predictive model. In the ppmr package, currently, only the bucket method based on the original trace clustering is implemented. In this method, first, calculate the distance between the preset traces. The optional distances include:

1. Optimal string alignment(restricted Damerau-Levenshtein distance)
2. Levenshtein distance
3. Full Damerau-Levenshtein distance
4. Hamming distance
5. Longest common substring distance
6. q-gram distance
7. cosine distance between q-gram profiles
8. Jaccard distance between q-gram profiles
9. aro-Winkler distance
10. Distance based on Soundex encoding

After calculating the distance, you can build a Hierarchical Cluster or a Kmeans Cluster. In the ppmr package, the Kmeans Cluster Bucketing function is `KMmeans ClusterBucketing`. The parameters of this function are:

1. prefix_eventLog: A event log with a prefix.
2. dist_methods: Measure the distance of the trace.

Table 3: Kmeans Cluster Bucketing

patient	handling	employee	handling_id	registration_type	time	cluster_id
1	Registration	r1	1	start	2017-01-02 11:41:53	1
1	Triage and Assessment	r2	501	start	2017-01-02 12:40:20	1
1	Registration	r1	1	complete	2017-01-02 12:40:20	1
1	Triage and Assessment	r2	501	complete	2017-01-02 22:32:25	1
1	Blood test	r3	1001	start	2017-01-05 08:59:04	1

Table 4: Hierarchical Cluster Bucketing

patient	handling	employee	registration_type	time	predicate	cluster_id
1	Registration	r1	start	2017-01-02 11:41:53	Discuss Results	1
1	Triage and Assessment	r2	start	2017-01-02 12:40:20	Discuss Results	1
1	Registration	r1	complete	2017-01-02 12:40:20	Discuss Results	1
1	Triage and Assessment	r2	complete	2017-01-02 22:32:25	Discuss Results	1
1	Blood test	r3	start	2017-01-05 08:59:04	Discuss Results	1

3. cluster_methods : Cluster methods. Options include Clara(Large data version of K-means), fanny(fuzzy K-means clustering), and pam(robust version of K-means).
4. cluster_num : Cluster number.

The following is an example of the Kmeans Cluster Bucketing function:

```
enrichEventlogEncoding <-
  KMmeanclusterBucketing(prefix_eventLog = eventdata)
```

The Hierarchical Cluster Bucketing function is `hierarchicalClusteringBucketing`. The parameters of this function are:

1. prefix_eventLog: A event log with a prefix.
2. dist_methods: Measure the distance of the trace.
3. cluster_methods : Cluster methods. Options include Agnes and Diana.
4. cluster_num : Cluster number.

The following is an example of the Hierarchical Cluster Bucketing function:

```
enrichEventlogEncoding <-
  hierarchicalClusteringBucketing(prefix_eventLog = eventdata)
```

Both methods return event logs with cluster tags In addition. You can also use the `optimalNumberOfCluster` function in the ppmr package to calculate the optimal number of clusters. This method uses Gap Statistics to Estimate the Number of Clusters.

Currently, the methods implemented in this package are based on the original prefix event log, which means that these methods only consider the information from the control flow perspective. Therefore, you can encode the prefix event log first and then calculate the distance between different traces for all features. This method has yet to be implemented.

5.0.3 Encoding prefix event log

All prefix tracking in the same bucket needs to be represented as a fixed-length feature vector to train the classifier. Three encoding methods are implemented in ppmr packet duplication, namely:

1. last state encoding.
2. last N state index-based encoding.
3. aggregation encoding.

Table 5: Last State Encoding

patient	handling	employee	registration_type	time	predicate
1	MRI SCAN	r4	complete	2017-01-06 01:54:23	Discuss Results
10	Discuss Results	r6	complete	2017-01-10 11:11:18	Check-out
100	Discuss Results	r6	complete	2017-04-18 01:11:09	Check-out
101	Discuss Results	r6	complete	2017-04-22 11:14:53	Check-out
102	Discuss Results	r6	complete	2017-04-22 08:15:23	Check-out

Table 6: Last N State Index Based Encoding

patient	handling_1	time_1	employee_1	handling_2	time_2
1	Triage and Assessment	2017-01-02 22:32:25	r2	Blood test	2017-01-05 14:34:27
10	Triage and Assessment	2017-01-08 08:33:09	r2	X-Ray	2017-01-09 19:29:09
100	Triage and Assessment	2017-04-12 03:46:08	r2	X-Ray	2017-04-12 18:34:04
101	Triage and Assessment	2017-04-17 10:17:27	r2	X-Ray	2017-04-18 05:22:16
102	Triage and Assessment	2017-04-16 22:46:00	r2	X-Ray	2017-04-18 01:05:58

`lastStateEncoding` function implements the last state encoding method. The function has only one parameter, namely the prefix eventLog. The example code is shown below:

```
enrichEventlogEncoding <-
  lastStateEncoding(prefix_eventLog = eventdata)
```

`lastNStateIndexBasedEncoding` function implements the last N State Index Based Encoding method. The function has two parameters:

1. Window: The Window length of the Trace.
2. eventLog: A event log with a prefix.

An example of the `lastNStateIndexBasedEncoding` function is shown below:

```
enrichEventlogEncoding <-
  lastNStateIndexBasedEncoding(prefix_eventLog = eventdata,
                                                    Window=3)
```

`aggregation_encoding` function implements the aggregation encoding method. An example of this method is shown below:

```
enrichEventlogEncoding <- aggregation_encoding(prefix_eventLog = eventdata)
```

Only these three coding methods are implemented in ppmr. Other techniques, such as tensor coding, still need to be implemented.

5.0.4 Build predictive model

Once you have defined the predicate in the event log and coded the event log, you can start to build the predictive model. As an example, consider a single bucket. That is, use all event logs to build predictive

Table 7: Aggregation Encoding

case_id	handling.Blood.test	handling.Check.out	handling.Discuss.Results	handling.MRI.SCAN
1	1	0	0	1
10	0	0	1	0
100	0	0	1	0
101	0	0	1	0
102	0	0	1	0

models. In the ppmr package, use the `BuildModel` function to construct the predictive model. The parameters of this function include the following:

1. TheModel: Model function in Tidymodels, such as boost_tree,rand_forest.
2. engine: engine of the model.
3. PrefixData: Encoding prefix event log data.
4. predictmode: Type of model, regression, or classification.

```
# data processing
EventLogModel <- BuildModel(TheModel = rand_forest,
                            engine = "ranger",
                            PrefixData = enrichEventlogEncoding.1,
                            predictmode = "classification")
# save(EventLogModel,file = "EventLogModel.RData")
```

Currently, the `BuildModel` function does not support parameter adjustment. Once the predictive model is trained, the effect of the model needs to be evaluated. The `evaluateModel` function in the ppmr package can be used to evaluate the built predictive model.

```
library(ppmr)
load("/Users/milin/Work/EventLogModel.Rdata")
x <- evaluateModel(ModelResult = EventLogModel[[3]],
                   train = EventLogModel[[1]],
                   test = EventLogModel[[2]],
                   predictmode = "classification")
x
```

```
## $train
##   precision    accuracy   sensitive specificity   f_measure
##   0.9982548   0.9973190   0.7500000   0.9986339   0.9991251
##
## $test
##   precision    accuracy   sensitive specificity   f_measure
##   0.9912281   0.9920000   0.6666667   0.9963768   0.9955752
```

We can go back to the previous steps if the model does not work well. If the model works well, we will go to the last step and try to understand the model.

5.0.5 Interpret Model

In the ppmr package, you can use the `InterpretiveModel` function to explain the model. This function has four parameters:

1. ModelResult: Model result from BuildModel function
2. train: Train data without the label.
3. trainlabel: Lable of the dataset.
4. newdata: Observations for prediction. This function will output a list containing four objects:
5. feature important
6. BD plot
7. Ceteris paribus Profiles
8. Ceteris paribus Oscillations

```
result <- InterpretativeModel(ModelResult = EventLogModel[[3]],
                              traindata=EventLogModel[[1]][,-3],
                              trainlabel = EventLogModel[[1]][,3],
                              newdata = EventLogModel[[2]][1,])
# save(result,file = "result.RData")
# feature important
plot(result[[1]])
```

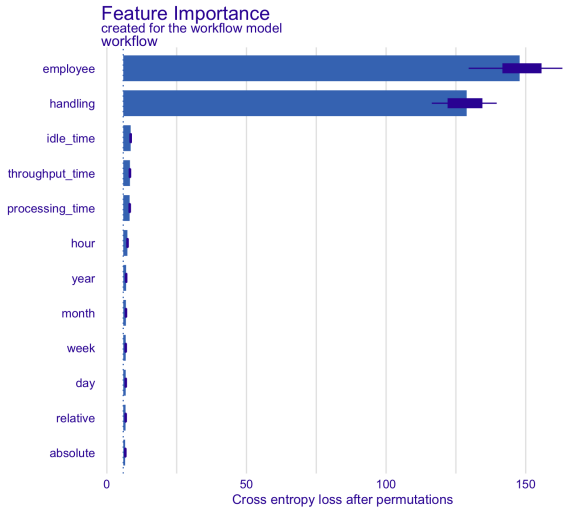

Figure 5: Feature Important

`InterpretativeModel` function returns a list of four elements. The first element in the list is the importance of the variable 5. From the results, we can see that the most critical variable is employee, followed by handling. The second element in the list is the break down plot.

```
# BDplot
plot(result[[2]])
```

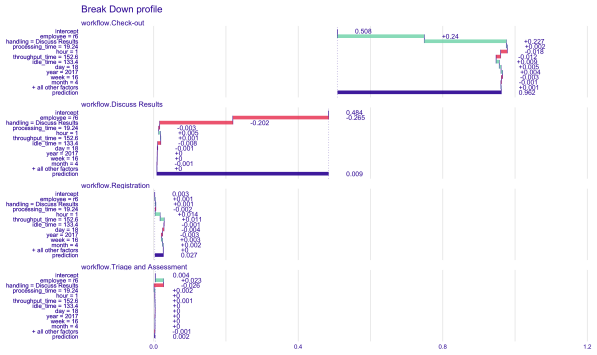

Figure 6: Break Down plot

Break Down plot 6 show how different variables affect the prediction results. The result of the third element in the list is Ceteris-paribus-Profiles 7, which shows how changes in variables affect predictions. The following analyzes how changes in processing time affect the prediction.

```
# Ceteris_paribus_Profiles
plot(result[[3]],variables = "processing_time")
```

The result of the fourth element is Ceteris paribus Oscillations 8, which also shows the importance of variables.

```
# Ceteris_paribus_Oscillations
plot(result[[4]])
```

Through this step, we can understand the important features in the model, which variables affect the prediction and the degree of impact, and how the model prediction will change if the input variables change. Furthermore, this information can help us understand the business and further optimize the business process.

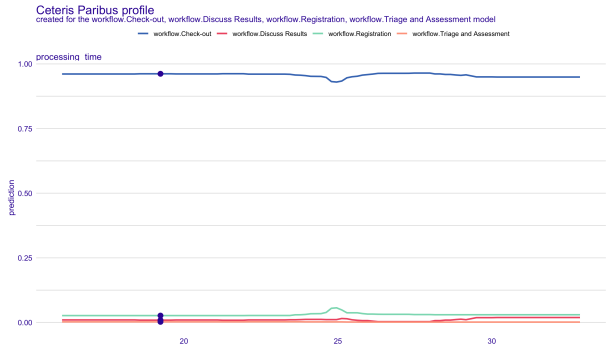

Figure 7: Ceteris Paribus Profiles

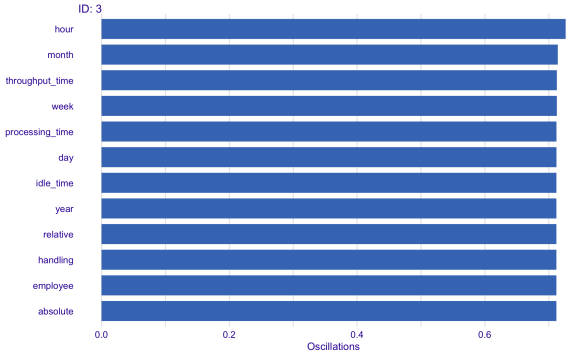

Figure 8: Ceteris Paribus Oscillations

6 Conclusion

Many researchers have proposed solutions to predictive process monitoring, but no researchers have proposed a framework to solve the problem of predictive process monitoring. Therefore, this paper tries to solve a problem. This paper divides the predictive process monitoring process into two stages: the training and prediction stages. In the training phase, there are seven steps:

1. Define the predictor of the event log.
2. Enrich the event log.
3. Prefix event log.
4. Encoding prefix event log.
5. Bucketing prefixes.
6. Train predictive model.
7. Interpret predictive Models.

The seven steps are somewhat random. For example, when the model is ineffective, you can try to return to the previous steps, such as using other encoding methods. In the prediction phase, there are four steps:

1. Prefix event log.
2. Encoding prefix event log.
3. Bucketing prefixes.
4. Apply predictive model.

In the prediction stage, the specific method selected in each step depends on the chosen method in the prediction stage. In addition, the technique of deep learning can also be applied under this framework. R and Python are the most popular data science languages at present. However, proM is the primary tool for process mining. There are few process mining tools in R and Python. There are only bupaR in R and pm4py library in Python. However, it mainly solves the problems in the two fields of process discovery and

compliance checking and needs to solve the tools of predictive process monitor. The development of the ppmr package fills this gap.

However, this paper only considers the whole process of proactive process mining, not the entire process of process mining. Future work can consider embedding the framework in the whole process mining process. The ppmr package needs to be improved. This package does not implement many methods, such as prefix length bucket methods. In addition, the functions in the ppmr package need to be faster, and the code needs to be optimized. This paper has yet to fully use the real event log data to conduct experiments to compare different predictive process monitoring methods under the analysis framework. This is also the next step of this paper.

Reference

Ahmed, Razi, Muhammad Faizan, and Anwer Irshad Burney. 2019. "Process Mining in Data Science: A Literature Review." In *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, 1–9. IEEE.

Alves de Medeiros, AK, and Wil MP van der Aalst. 2008. "Process Mining Towards Semantics." In *Advances in Web Semantics i*, 35–80. Springer.

Appice, Annalisa, Nicola Di Mauro, and Donato Malerba. 2019. "Leveraging Shallow Machine Learning to Predict Business Process Behavior." In *2019 IEEE International Conference on Services Computing (SCC)*, 184–88. IEEE.

Archer, Kellie J, and Ryan V Kimes. 2008. "Empirical Characterization of Random Forest Variable Importance Measures." *Computational Statistics & Data Analysis* 52 (4): 2249–60.

Bergenthum, Robin, Jörg Desel, Robert Lorenz, and Sebastian Mauser. 2007. "Process Mining Based on Regions of Languages." In *International Conference on Business Process Management*, 375–83. Springer.

Bernard, Gaël, and Periklis Andritsos. 2019. "Accurate and Transparent Path Prediction Using Process Mining." In *European Conference on Advances in Databases and Information Systems*, 235–50. Springer.

Berti, Alessandro, Sebastiaan J Van Zelst, and Wil van der Aalst. 2019. "Process Mining for Python (PM4Py): Bridging the Gap Between Process-and Data Science." *arXiv Preprint arXiv:1905.06169*.

Biecek, Przemysław. 2018. "DALEX: Explainers for Complex Predictive Models in r." *The Journal of Machine Learning Research* 19 (1): 3245–49.

Broucke, Seppe KLM vanden, and Jochen De Weerdt. 2017. "Fodina: A Robust and Flexible Heuristic Process Discovery Technique." *Decision Support Systems* 100: 109–18.

Ceci, Michelangelo, Pasqua Fabiana Lanotte, Fabio Fumarola, Dario Pietro Cavallo, and Donato Malerba. 2014. "Completion Time and Next Activity Prediction of Processes Using Sequential Pattern Mining." In *International Conference on Discovery Science*, 49–61. Springer.

Dakic, Dusanka, Srdjan Sladojevic, Teodora Lolic, and Darko Stefanovic. 2019. "Process Mining Possibilities and Challenges: A Case Study." In *2019 IEEE 17th International Symposium on Intelligent Systems and Informatics (SISY)*, 000161–66. IEEE.

De Leoni, Massimiliano, Wil MP van der Aalst, and Marcus Dees. 2016. "A General Process Mining Framework for Correlating, Predicting and Clustering Dynamic Behavior Based on Event Logs." *Information Systems* 56: 235–57.

De Smedt, Johannes, Jochen De Weerdt, and Jan Vanthienen. 2014. "Multi-Paradigm Process Mining: Retrieving Better Models by Combining Rules and Sequences." In *OTM*, 446–53. Springer.

De Weerdt, Jochen, Manu De Backer, Jan Vanthienen, and Bart Baesens. 2012. "A Multi-Dimensional Quality Assessment of State-of-the-Art Process Discovery Algorithms Using Real-Life Event Logs." *Information Systems* 37 (7): 654–76.

Di Francescomarino, Chiara, Marlon Dumas, Fabrizio Maria Maggi, and Irene Teinemaa. 2016. "Clustering-Based Predictive Process Monitoring." *IEEE Transactions on Services Computing* 12 (6): 896–909.

Dongen, Boudewijn F van, Ronald A Crooy, and Wil MP van der Aalst. 2008. "Cycle Time Prediction: When Will This Case Finally Be Finished?" In *OTM Confederated International Conferences" on the Move to Meaningful Internet Systems"*, 319–36. Springer.

Elkhawaga, Ghada, Mervat Abu-Elkheir, and Manfred Reichert. 2022. "Explainability of Predictive Process Monitoring Results: Can You See My Data Issues?" *Applied Sciences* 12 (16): 8192.

Evermann, Joerg, Jana-Rebecca Rehse, and Peter Fettke. 2016. "A Deep Learning Approach for Predicting Process Behaviour at Runtime." In *International Conference on Business Process Management*, 327–38. Springer.

———. 2017. "Predicting Process Behaviour Using Deep Learning." *Decision Support Systems* 100: 129–40.

Ferilli, Stefano, and Sergio Angelastro. 2019. "Activity Prediction in Process Mining Using the WoMan Framework." *Journal of Intelligent Information Systems* 53 (1): 93–112.

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2019. "All Models Are Wrong, but Many Are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously." *J. Mach. Learn. Res.* 20 (177): 1–81.

Folino, Francesco, Massimo Guarascio, and Luigi Pontieri. 2012. "Discovering Context-Aware Models for Predicting Business Process Performances." In *OTM Confederated International Conferences" on the Move to Meaningful Internet Systems"*, 287–304. Springer.

———. 2013. "Discovering High-Level Performance Models for Ticket Resolution Processes." In *OTM Confederated International Conferences" on the Move to Meaningful Internet Systems"*, 275–82. Springer.

Ghattas, Johny, Pnina Soffer, and Mor Peleg. 2014. "Improving Business Process Decision Making Based on Past Experience." *Decision Support Systems* 59: 93–107.

Grömping, Ulrike. 2007. "Relative Importance for Linear Regression in r: The Package Relaimpo." *Journal of Statistical Software* 17: 1–27.

Günther, Christian W, and Wil MP Van Der Aalst. 2007. "Fuzzy Mining–Adaptive Process Simplification Based on Multi-Perspective Metrics." In *International Conference on Business Process Management*, 328–43. Springer.

Jansen-Vullers, Monique H, Wil MP van der Aalst, and Michael Rosemann. 2006. "Mining Configurable Enterprise Information Systems." *Data & Knowledge Engineering* 56 (3): 195–244.

Janssenswillen, Gert, Benoit Depaire, Marijke Swennen, Mieke Jans, and Koen Vanhoof. 2019. "bupaR: Enabling Reproducible Business Process Analysis." *Knowledge-Based Systems* 163: 927–30.

Kozak, Anna. 2020. "Local Variable Importance via Oscillations of Ceteris Paribus Profiles." PhD thesis, Zakład Projektowania Systemów CAD/CAM i Komputerowego Wspomagania Medycyny.

Kozak, Anna, and Przemysław Biecek. n.d. "Vivo: Local Variable Importance via Oscillations."

Kursa, Miron B, Aleksander Jankowski, and Witold R Rudnicki. 2010. "Boruta–a System for Feature Selection." *Fundamenta Informaticae* 101 (4): 271–85.

Lakshmanan, Geetika T, Davood Shamsi, Yurdaer N Doganata, Merve Unuvar, and Rania Khalaf. 2015. "A Markov Prediction Model for Data-Driven Semi-Structured Business Processes." *Knowledge and Information Systems* 42 (1): 97–126.

Lamma, Evelina, Paola Mello, Marco Montali, Fabrizio Riguzzi, and Sergio Storari. 2007. "Inducing Declarative Logic-Based Models from Labeled Traces." In *International Conference on Business Process Management*, 344–59. Springer.

Larose, Chantal D, and Daniel T Larose. 2019. *Data Science Using Python and r*. John Wiley & Sons.

Leemans, Sander JJ, Dirk Fahland, and Wil MP Van Der Aalst. 2014. "Process and Deviation Exploration with Inductive Visual Miner." *BPM (Demos)* 1295 (8).

Leoni, Massimiliano de, Wil MP Van der Aalst, and Marcus Dees. 2014. "A General Framework for Correlating Business Process Characteristics." In *International Conference on Business Process Management*, 250–66. Springer.

Leontjeva, Anna, Raffaele Conforti, Chiara Di Francescomarino, Marlon Dumas, and Fabrizio Maria Maggi. 2016. "Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes." In *International Conference on Business Process Management*, 297–313. Springer.

Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems* 30.

Maggi, Fabrizio Maria, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. 2014. "Predictive Monitoring of Business Processes." In *International Conference on Advanced Information Systems Engineering*, 457–72. Springer.

Márquez-Chamorro, Alfonso E, Manuel Resinas, Antonio Ruiz-Cortés, and Miguel Toro. 2017. "Run-Time Prediction of Business Process Indicators Using Evolutionary Decision Rules." *Expert Systems with Applications* 87: 1–14.

Mehdiyev, Nijat, Joerg Evermann, and Peter Fettke. 2017. "A Multi-Stage Deep Learning Approach for Business Process Event Prediction." In *2017 IEEE 19th Conference on Business Informatics (CBI)*, 1:119–28. IEEE.

Molnar, Christoph, Gunnar König, Bernd Bischl, and Giuseppe Casalicchio. 2020. "Model-Agnostic Feature Importance and Effects with Dependent Features–a Conditional Subgroup Approach." *arXiv Preprint arXiv:2006.04628*.

Navarin, Nicolo, Beatrice Vincenzi, Mirko Polato, and Alessandro Sperduti. 2017. "LSTM Networks for Data-Aware Remaining Time Prediction of Business Process Instances." In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–7. IEEE.

Pandey, Suraj, Surya Nepal, and Shiping Chen. 2011. "A Test-Bed for the Evaluation of Business Process Prediction Techniques." In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 382–91. IEEE.

Polato, Mirko, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. 2014. "Data-Aware Remaining Time Prediction of Business Process Instances." In *2014 International Joint Conference on Neural Networks (IJCNN)*, 816–23. IEEE.

Polato, Mirko, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. 2018. "Time and Activity Sequence Prediction of Business Process Instances." *Computing* 100 (9): 1005–31.

Rogge-Solti, Andreas, and Mathias Weske. 2013. "Prediction of Remaining Service Execution Time Using Stochastic Petri Nets with Arbitrary Firing Delays." In *International Conference on Service-Oriented Computing*, 389–403. Springer.

———. 2015. "Prediction of Business Process Durations Using Non-Markovian Stochastic Petri Nets." *Information Systems* 54: 1–14.

Rozinat, Anne, Ivo SM de Jong, Christian W Günther, and Wil MP van der Aalst. 2009. "Process Mining Applied to the Test Process of Wafer Scanners in ASML." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39 (4): 474–79.

Santos Garcia, Cleiton dos, Alex Meincheim, Elio Ribeiro Faria Junior, Marcelo Rosano Dallagassa, Denise Maria Vecino Sato, Deborah Ribeiro Carvalho, Eduardo Alves Portela Santos, and Edson Emilio Scalabrin. 2019. "Process Mining Techniques and Applications–a Systematic Mapping Study." *Expert Systems with Applications* 133: 260–95.

Senderovich, Arik, Chiara Di Francescomarino, Chiara Ghidini, Kerwin Jorbina, and Fabrizio Maria Maggi. 2017. "Intra and Inter-Case Features in Predictive Process Monitoring: A Tale of Two Dimensions." In *International Conference on Business Process Management*, 306–23. Springer.

Sikaroudi, Amir Mohammad Esmaieeli, and Md Habibor Rahman. 2020. "Predictive Process Mining by Network of Classifiers and Clusterers: The PEDF Model." *arXiv Preprint arXiv:2011.11136*.

Spree, Florian. 2020. "Predictive Process Monitoring-a Use-Case-Driven Literature Review." In *EMISA Forum: Vol. 40, No. 1.* De Gruyter.

Staniak, Mateusz, and Przemyslaw Biecek. 2018. "Explanations of Model Predictions with Live and breakDown Packages." *arXiv Preprint arXiv:1804.01955*.

Tax, Niek, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. 2017. "Predictive Business Process Monitoring with LSTM Neural Networks." In *International Conference on Advanced Information Systems Engineering*, 477–92. Springer.

Teinemaa, Irene, Marlon Dumas, Fabrizio Maria Maggi, and Chiara Di Francescomarino. 2016. "Predictive Business Process Monitoring with Structured and Unstructured Data." In *International Conference on Business Process Management*, 401–17. Springer.

Theis, Julian, and Houshang Darabi. 2019. "Decay Replay Mining to Predict Next Process Events." *IEEE Access* 7: 119787–803.

Van Der Aalst, Wil. 2016. *Process Mining: Data Science in Action*. Vol. 2. Springer.

Van der Aalst, Wil MP, and Boudewijn F van Dongen. 2002. "Discovering Workflow Performance Models from Timed Logs." In *International Conference on Engineering and Employment of Cooperative Information Systems*, 45–63. Springer.

Van der Aalst, Wil MP, M Helen Schonenberg, and Minseok Song. 2011. "Time Prediction Based on Process Mining." *Information Systems* 36 (2): 450–75.

Van der Aalst, Wil MP, and Minseok Song. 2004. "Mining Social Networks: Uncovering Interaction Patterns in Business Processes." In *International Conference on Business Process Management*, 244–60. Springer.

Van der Aalst, Wil MP, Boudewijn F Van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and Anton JMM Weijters. 2003. "Workflow Mining: A Survey of Issues and Approaches." *Data & Knowledge Engineering* 47 (2): 237–67.

Van der Aalst, Wil MP, and Anton JMM Weijters. 2004. "Process Mining: A Research Agenda." *Computers in Industry* 53 (3): 231–44.

Van Dongen, Boudewijn F, Ana Karla A de Medeiros, HMW Verbeek, AJMM Weijters, and Wil MP van Der Aalst. 2005. "The ProM Framework: A New Era in Process Mining Tool Support." In *International Conference on Application and Theory of Petri Nets*, 444–54. Springer.

Vasilyev, Evgeniy, Diogo R Ferreira, and Junichi Iijima. 2013. "Using Inductive Reasoning to Find the Cause of Process Delays." In *2013 IEEE 15th Conference on Business Informatics*, 242–49. IEEE.

Velmurugan, Mythreyi, Chun Ouyang, Catarina Moreira, and Renuka Sindhgatta. 2021. "Evaluating Stability of Post-Hoc Explanations for Business Process Predictions." In *International Conference on Service-Oriented Computing*, 49–64. Springer.

Verenich, Ilya, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Chiara Di Francescomarino. 2016. "Complex Symbolic Sequence Clustering and Multiple Classifiers for Predictive Process Monitoring." In *International Conference on Business Process Management*, 218–29. Springer.

Verenich, Ilya, Stanislav Moškovski, Simon Raboczi, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. 2018. "Predictive Process Monitoring in Apromore." In *International Conference on Advanced Information Systems Engineering*, 244–53. Springer.

Zandkarimi, Fareed, Jana-Rebecca Rehse, Pouya Soudmand, and Hartmut Hoehle. 2020. "A Generic Framework for Trace Clustering in Process Mining." In *2020 2nd International Conference on Process Mining (ICPM)*, 177–84. IEEE.