Decision Mining with Time Series Data Based on Automatic Feature Generation

Beate Scheibel[1][0000−0002−1513−199X] and Stefanie Rinderle-Ma[2][0000−0001−5656−6108]

[1] Research Group Workflow Systems and Technology, Faculty of Computer Science, University of Vienna
beate.scheibel@univie.ac.at
[2] Chair of Information Systems and Business Process Management, Department of Informatics, Technical University of Munich, Germany
stefanie.rinderle-ma@tum.de

Abstract. Decision rules play a crucial role in business process execution. Knowing and understanding decision rules is of utmost importance for business process analysis and optimization. So far, decision discovery has been merely based on data elements that are measured at a single point in time. However, as cases from different application areas show, process behavior and process outcomes might be heavily influenced by additional data such as sensor streams, that consist of time series data. This holds also true for decision rules based on time series data such as 'if temperature > 25 for more than 3 times, discard goods'. Hence, this paper analyzes how time series data can be automatically exploited for decision mining, i.e., for discovering decision rules based on time series data. The paper identifies global features as well as patterns and intervals in time series as relevant for decision mining. In addition to global features, the paper proposes two algorithms for discovering interval-based and pattern-based features. The approach is implemented and evaluated based on an artificial data set as well as on a real-world data set from manufacturing. The results are promising: the approach discovers decision rules with time series features with high accuracy and precision.

Keywords: Decision Mining · Time Series Data · Process-Aware Information Systems · Process Mining · Process Analysis

1 Introduction

Process mining encompasses process discovery, conformance checking, and process enhancement [1]. An important aspect of process discovery is decision mining, which focuses on discovering decision points in processes and the underlying decision rules based on event logs [15]. Existing decision mining algorithms detect decision rules including data elements [13], overlapping rules [16] as well as incorporating linear relationships between variables [14].

However, these approaches do not take into account time series data[3]. Time series data is especially relevant as many application domains collect context data outside the process as well as inside the process in the form of time series data, i.e. sensors, such as in [7], that might influence the process behavior to a great extent [5], e.g., causing concept drifts [23] and driving decisions [6].

Use cases for discovering decision rules based on time series data exist in different domains. In healthcare, for example, blood values may be decisive for the further treatment of the patient. However, not the last blood value alone might be important, but the overall trend of the blood samples, i.e., was the value decreasing or increasing over time. Another example which will be used as running example throughout the paper stems from the logistics domain and is loosely based on the use case mentioned in [6]. The corresponding process model can be seen in Fig. 1. Temperature sensitive cargo is loaded onto a transporter and moved to a destination, where the cargo is unloaded and transferred to the customer. During the transportation, the temperature is measured 15 times[4]. As the transporter reaches the destination, it is checked if the temperature exceeded 25 degrees for more than three times. If that was the case, the goods are 'NOK' and have to be discarded, otherwise they will be transferred to the customer.

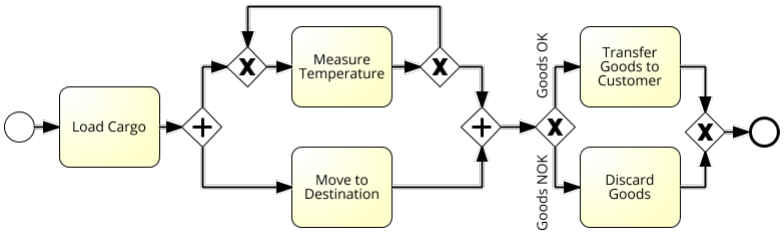

Fig. 1. BPMN model of logistics use case (modeled using Signavio©).

The corresponding decision rule looks like follows:

Rule Running Example:
IF temperature $>$ 25 FOR number_measurements $>$ 3 THEN discard goods.

Despite the high relevance of contextual data for process behavior [5, 4, 22, 23, 7], only [6] has addressed decision rule discovery based on time series data

[3] Time series are defined as a sequence of time-stamped, or at least ordered, data with real-valued attribute values. Note that in this paper, we do not assume equidistant observation times.

[4] Note, that the 'Measure Temperature' task is modelled explicitly here for illustration purposes. However, it could also stem from an external source.

so far, but in an interactive, non-automated way. Hence, this work tackles the following research question:

RQ: *How to discover decision rules based on time series data with high accuracy and high precision in an automated way?*

When exploiting time series data in decision mining approaches, it is important to achieve high accuracy and precision for the extracted rules, i.e., the goal is to discover rules that represent the underlying ground truth as closely as possible. This is important to achieve validity and robustness of the discovered rules as well as to provide interpretable, expressive rules, and transparency [20].

To answer ***RQ***, we first derive three decision rule patterns based on an analysis of time series patterns and existing classification techniques from literature and use cases. In addition to the provision of necessary preprocessing steps, this paper contributes algorithms for the discovery of the derived decision rule patterns based on global and interval-based features as well as pattern-based features. The output are textual decision rules that take time series data into account. The overall approach is prototypically implemented and evaluated on synthetic data of the running example and real-world data from the manufacturing domain, i.e., production of a workpiece with accompanying sensor data. On both data sets, the approach yields decision rules with time series data at high accuracy and precision.

Section 2 features three decision rule patterns with time series data based on a literature analysis and use cases. Section 3 exploits the results of this analysis to provide an approach for discovering time series based decision rules, which is evaluated in Sect. 4 and discussed in Sect. 5. An overview of related work is given in Sect. 6 and a conclusion is provided in Sect. 7

2 Time Series Based Decision Rules - Analysis

For building the basis for discovering decision rules with time series data, this section analyzes literature on time series patterns and classification of time series data, with focus on expressive, interpretable decision rules. Interpretability is an important aspect in decision mining to provide transparent and explainable results [12]. The results of this analysis allow for the definition of three time series dependent decision rule patterns.

Time Series Patterns: Time series are classified into different categories, e.g., discrete or continuous time series, univariate or multivariate time series [11, 17]. For this paper we focus on discrete, univariate time series where separate measurements are recorded on specific points in time. Multivariate time series, i.e., multiple time series, potentially influencing each other, will be part of future work. Especially interesting are time series patterns, as these might be insightful for underlying decision rules. In literature, different time series patterns are defined: stationary, random fluctuations, trends, level shifts, periods/cycles/seasonal variations or combinations of patterns [17]. Looking at process mining use cases, as well as the running example in Sect. 1, we add 'thresholds' as a time

series pattern. Thresholds can occur once, which is a straightforward condition, or have to be met a certain amount of times to be decision relevant.

This leads to the following comprehensive, but not complete, list of decision relevant patterns: *Stationarity, Trends, Periods/Cycles/Seasonal Variations, Shifts, Thresholds, and Pattern Combinations.*, cf. Fig. 2.

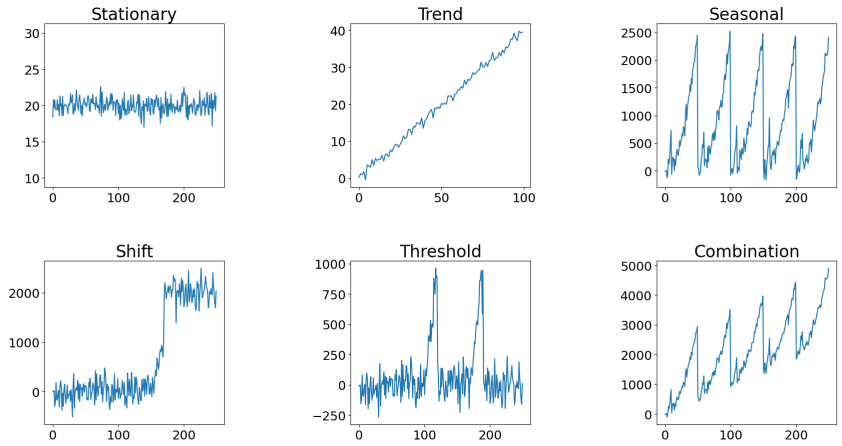

Fig. 2. Time Series Patterns.

Stationarity refers to a time series, where the mean stays constant over time. A *Trend* is defined by an increasing or decreasing mean value. Both of these patterns follow a linear trend. *Periods/Cycles/Seasonal Variations* are similar in that the values repeatedly fluctuate over a specific time span. There are differences, for example, cycles usually refer to longer time spans than seasonal variations. However, for this paper, it is seen as one category, i.e., *Seasonal*, as they can be analyzed similarly. If a sudden, but lasting increase or decrease occurs in the recorded values or the mean changes abruptly, a *Shift* was detected. In contrast, *Thresholds* refer to sudden, non-lasting increases or decreases. Different *Combinations* of these patterns and combinations of time series dependent rules and other decision rules can occur as well.

Existing Approaches for Time Series Data Classification: Decision mining can be understood as a classification problem, as the path that a particular process instance takes can be seen as a category and the decision rule as classifier [15]. Hence, we regard approaches for classification of time series data as suitable input for our further considerations. The simplest time series classification approach is to treat each value of time series as a separate feature. However, valuable information might be lost. A common approach is distance-based classification, i.e. calculating the distance between several time series and clustering them based on the calculated distance [2]. However, for decision mining, we do

not want to classify instances into the right category, but extract the distinguishing features to formulate textual decision rules.

Other approaches focus on extracting characteristic features from time series data to classify the instances. Which kind of features are used varies and can be calculated either on the whole time series or only on specific parts [9]. The generation of these features leads to a reduction in dimensionality which enables the application of already existing classification algorithms. As additional features enable interpretability as well as the application of existing decision mining algorithms, these approaches seem more applicable. [3] gives a comprehensive overview of existing classification algorithms for time series data. These algorithms are classified into six categories: time domain distance-based classifiers, differential distance-based classifiers, dictionary-based classifiers, shapelet-based classifiers, interval-based classifiers and ensemble classifiers. The first two classifiers fall into the distance-based category and are therefore not relevant for our purpose. Shapelet-based classifiers try to identify subsequences in a time series, that are decisive for the classification. Dictionary-based classifiers, transform time series into representative words and subsequently compare the distribution of words. Both of the latter approaches therefore explicitly take the distribution of values into account. Interval-based classifiers, split the time series into intervals and calculate different features on top of these intervals. Ensemble classifiers combine the previously described approaches. In addition, several deep learning methods [8] have been proposed. However, as we want to generate human readable, interpretable rules [20], we need interpretable approaches, i.e., standard deep learning methods are not applicable. This also applies to ensemble classifiers, as they do not provide the decisive features. Shapelet-based approaches allow to visualize the important subsequences of time series. However, they also do not allow for the extraction of expressive, textual decision rules.

Therefore, feature-based approaches, e.g., dictionary or interval-based classifiers, are suitable for decision mining, as they enable the generation of interpretable decision rules. Following the described techniques, the features can be generated on the entire time series or on specific intervals, also including the specific distribution of values.

Decision Rule Patterns: The description of patterns in combination with the described techniques allows to define some discriminating features that can be used to search for decision relevant patterns. For example, both a trend as well as stationary data can be defined by the overall slope or the overall percentage change. However, this is not applicable for the other patterns, as for example in a seasonal pattern, the overall slope is not informative, but rather the slope of parts, i.e., intervals of the time series. In a threshold pattern, we might rather look at the distribution of individual values, compare the running example, where specific values and their number of occurrence are the decisive characteristic. To discover combinations of patterns, combinations of these features might be necessary. These observations lead to the definition of decision rule patterns 2 - 4 in comparison to the baseline decision rule pattern 1, that can be typically seen in decision rules.

Decision Rule Pattern 1 (Baseline Decision Rule)

$$IF\ v(ariable)\ op(erator)\ c(onstant)\ THEN\ class\_x$$

with $op \in \{<, >, \leq, \geq, \neq, =\}$ *and* $c \in \mathbb{R}$

An example for a baseline decision rule is 'IF measurement_x > 2.5 THEN NOK'.

Decision Rule Pattern 2 (Global feature-based decision rule)

$$IF\ global\_feature(v)\ op\ c\ THEN\ class\_y$$

Global features are summary features, that can be calculated over the entire time series. This rule pattern will be especially useful if the underlying time series is either stationary or includes a trend.

Decision Rule Pattern 3 (Interval-based decision rule)

$$IF\ (feature(v)\ in\ interval\_n)\ op\ c\ THEN\ class\_x$$

Interval-based features, refer to the same features as the global feature, but these calculations are applied on individual intervals instead of the entire time series.

Decision Rule Pattern 4 (Pattern-based decision rule)

$$IF\ v\ op\ c\ FOR\ \{n\_times, timerange\}\ THEN\ class\_x$$

Pattern-based features take into account the distribution of values in a time series. This can be applied globally or on each interval of a time series.

Decision Rule Patterns 3 and 4 are mostly used for more complex time series, i.e., ones with seasonal variations, shifts, thresholds, or combinations of patterns. In general, decision rules can be based upon the presence or absence of a particular time series pattern.

3 Approach - EDT-TS

Based on Decision Rule Patterns 2 – 4 (cf. Sect. 2), we propose the *Extended Decision Tree - Time Series (EDT-TS)* approach. We base the EDT-TS approach on a decision tree as this enables the generation of interpretable rules. To allow the integration of time series data, additional features are generated, according to the analysis of decisive features in Sect. 2.

The approach involves a preprocessing stage, a feature generation stage, where additional features are created using three techniques as well as a rule extraction stage, where the actual discovery of decision rules takes place. These stages are described in more detail in the following subsections.

3.1 Preprocessing

In this paper, we assume that the time series data is part of an event log, either in form of separate, repeated tasks with one value each (cf. Tab.1) or one task with a list of measurement data (cf. Tab.2). Other ways of integrating sensor data into event logs are conceivable, such as aggregating sensor data based on task annotations [7], but outside the scope of this paper. For preprocessing the event logs are read and converted into a tabular form, specifically a Dataframe[5], where each row refers to one instance. To specify the candidate variables for time series analysis, all reoccurring, numeric variables are identified, these are then used as input for the feature generation. The preprocessing step has to be adapted for different use cases, as different actions have to be performed to convert an event log into a suitable dataframe. However, from this point on, the process remains the same for multiple use cases.

Table 1. Event log including measurements in one task.

UUID	Task	Data
0001	Measure	10,15,14,16,10,10,14,12,14
0001	Measure	12,13,14,12,10,11,13,15,12

Table 2. Measurements as separate tasks.

UUID	Task	Timestamp	Data
0001	Measure	2019-11-15 14:35	10
0002	Measure	2019-11-15 16:40	12
0001	Measure	2019-11-15 14:45	15
0001	Measure	2019-11-15 14:52	14
0002	Measure	2019-11-15 16:55	13
0002	Measure	2019-11-15 17:10	14

3.2 Feature Generation

After the preprocessing step, additional features are generated.

Global features are calculated for the whole time series. They can consist of simple values, for example the mean, variance, slope, percentage change or the number of peaks/lows in the time series. In addition, more complex values can be computed, e.g., a Fourier transform or auto-regressive coefficients. These global features are calculated for the time series data of each instance and added as additional features.

For **interval-based features**, the time series is split into intervals and global features are calculated for each interval. Figure 3 depicts the first interval and the n-th interval of a time series where features like the mean or percentage change vary greatly.

The split of the time series can be done according to measurement points (as in this paper) or time spans. It is important to set the interval size appropriately, as this can have an effect on the resulting decision rules as well as the time

[5] `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html`

complexity of the algorithm. For this implementation we chose to split the time series into 2, 5, and 10 intervals. However, the interval size can also be chosen manually for use cases where other interval sizes may be optimal. The generation of global features is done using an existing library, the generation of interval-based features can be seen in Alg. 1.

Algorithm 1 EDT-TS, interval-based features

Input: event log as dataframe, candidates, number of intervals n
Output: dataframe with added generated features

```
for instance in event log do
    split candidates into n intervals
end for
for interval in intervals do
    for feature in features(mean, minimum, maximum, slope,...) do
        calculate feature
        add result as additional feature
    end for
end for
return dataframe with added columns, i.e. the generated features
```

Pattern-based features consider the distribution of values in a time series. For these features, the actual values and their number of occurrence is important. This enables for example the discovery of threshold patterns as we can identify values that occur significantly more often in one class than in the other.

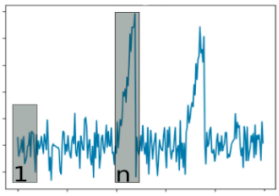

Fig. 3. Time series intervals.

Different algorithms and approaches can be subsumed under pattern-based approaches and there exists a variety of already implemented algorithms, e.g., dictionary-based approaches. However, dictionary-based approaches combine multiple values into one letter or word and therefore loose some informative value in favor of computational complexity. In addition, existing algorithms often do not enable the extraction of the original feature-value distribution, before it was converted to a word which makes it harder to extract textual rules. Therefore, Alg. 2 is proposed to identify possible thresholds of values based on the distribution of values in series. Note that Alg. 2 currently works on the whole time series. However, it can also be easily adapted to work on intervals which allows to discover even more fine-grained patterns.

Algorithm 2 EDT-TS, pattern-based features

Input: dataframe, candidates, output from Alg.1
Output: dataframe with added generated features

split instances by the result category, i.e. in 'OK' and 'NOK'
for category in categories **do**
    calculate all values and number of occurrence in candidate variables
    store in array
end for
compare value distributions of categories
store all values distributions (measurement, amount) that only occur in one category in array *candidate_threshold*
for c in candidate_threshold **do**
    add c as new column in dataframe
    for i in instance **do**
        if c.measurement occurs in i more or equal times than c.amount **then**
            set value for c of i "True"
        else
            set value for c of i "False"
        end if
    end for
end for
return dataframe with added columns, i.e. the generated features

3.3 Rule Extraction

After the additional features are created, a decision tree is applied to build the decision rules. The decision tree parameters are optimized to provide the best precision values, i.e., splits in different classes. Feature selection is used to obtain a maintainable amount of features. The last step is to output the decision tree results in human readable, textual form. This is done by following the nodes of the decision tree down for each individual class and concatenating the conditions to formulate a rule. Therefore multiple decision rules can be obtained, if different condition combinations can lead to the same class.

4 Evaluation

The approach has been implemented as a proof-of-concept prototype using Python. To generate the global features the 'tsfresh' module[6] is used, that allows to automatically create global features of a time series. To generate the decision rules, a decision tree from the 'scikit-learn' module [18] is used. In addition to the implementation of Alg. 1 and 2, a script was written to obtain textual rules from the resulting decision tree. The implemented approach was tested on two datasets, a synthetic dataset and a real-life dataset from the manufacturing domain. The decision tree implementation is set to aim for optimal

[6] `https://tsfresh.readthedocs.io/`

precision values. To enable comparison of the results, baseline values are created as well as individual results for each of the feature generation approaches. The source code, the datasets as well as the full results are available online: `https://github.com/bscheibel/edt-ts`. For the *baseline accuracy*, decision rules are calculated using a standard decision mining approach, without a component that is able to handle time series data. Therefore, only the individual values are used as input for this approach. The baseline accuracy is calculated to compare the results of 'standard' approaches, to EDT-TS. For the calculation, a 'CART' implementation from the 'scikit-learn' module was used, as this is a standard tool for decision mining [15]. The baseline results can be seen under 'Decision Rule, Baseline'.

The generated decision rule for each of the feature generation approaches (global features, interval-based and pattern-based) can be seen in the full result report online, here only the resulting, combined decision rule is shown. To test the accuracy, the data was split into a test and training set (80% training, 20% test). To calculate the accuracy, the following definition is used:

$$Accuracy := \frac{Number\ of\ correctly\ classified\ instances}{Total\ number\ of\ instances}$$

In addition to the accuracy, the precision of the result is calculated as well, using the following definition:

$$Precision := \frac{Total\ number\ of\ instances\ correctly\ classified\ in\ category}{Total\ number\ of\ instances\ classified\ as\ that\ category}$$

This definition leads to a precision value for each category, i.e., for 'OK' and 'NOK'. The depicted rules for both datasets, only contain the rules for the class 'NOK', as we assume that all instances that do not belong to 'NOK' are automatically classified as 'OK'. Accuracy and precision are choosen as evaluation metrics, as accuracy enables an intuitive assessment of the ability to identify correctly classified instances and precision, especially precision per class, provides information if the classification is imbalanced towards one class. This is especially relevant for the 'Manufacturing Dataset', as it is preferable that all 'NOK' pieces are detected early on. However it is more important that no 'OK' pieces should be wrongly disposed.

Running Example For the running example, synthetic data was generated. In total 5000 instances for the process model shown in Fig. 1 were created, where about 50% of these instances are 'OK' (2589 instances), i.e. the cargo can be transferred to the customer, and 50% are 'NOK' (2411 instances). The temperature was randomly generated (values between 10 and 30) for 15 measurement points for each instance. Figure 4 shows the measurement time series data for two instances. It is noticeable that the 'NOK' instance reaches higher temperatures more often. However no explicit decision rule could be derived from this figure alone. The result for the baseline approach can be seen below. The feature 'temperaturelast' refers to the last temperature measurement.

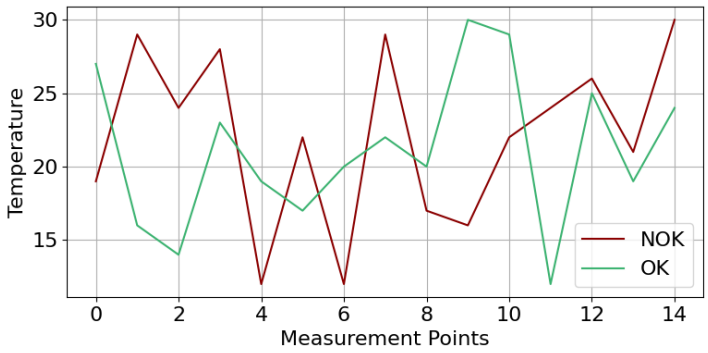

Fig. 4. Running example: visualisation of two instances. Red: 'Discard Goods'/'NOK' , Green: 'Transfer Goods'/'OK'.

The resulting decision tree contains 41 nodes in total, however most conditions are redundant, therefore the depicted decision rule was simplified. The precision of 59% for the class 'NOK' can be interpreted as 59% of all instances classified as 'NOK' are actually 'NOK'. The same applies for the 'OK' precision with a ratio of 56%.

Rule Running Example, Baseline:
If temperaturelast $>$ 25.50 THEN class: NOK/Discard Goods
Accuracy: 59%
Precision: NOK - 68%, OK - 56%

Rule Running Example, EDT-TS
If temperaturelist.count(26.0)$>=$ 4.0 $==$ *True* THEN class: NOK/Discard Goods
Accuracy: 100%
Precision: NOK - 100%, OK- 100%

The resulting rule contains a pattern-based feature, i.e., the temperature value of 26 has to occur equal or more than 4 times, which accurately represents the underlying rule. This reflects in the high accuracy and precision values.

Manufacturing Dataset To evaluate the applicability of the presented approach on a real world dataset, a manufacturing dataset from the production of valve lifters for gas turbines, see Fig. 5, is used. This dataset is an extension of the data used in [23] and was also used in [7].
Figure 6 shows the process model. Workpieces, i.e., the valve lifters are produced using a turning machine. Subsequently,

Fig. 5. Valve Lifter.

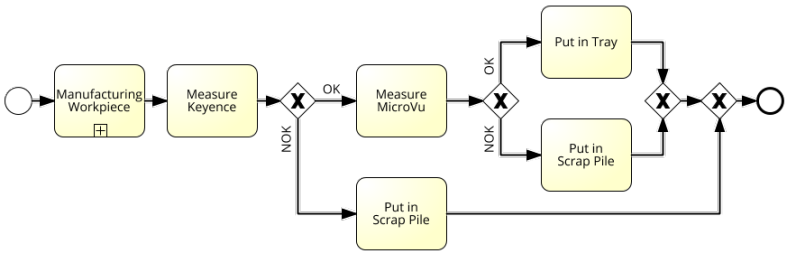

Fig. 6. Manufacturing processes (modeled using Signavio©).

the diameter of the workpieces is measured using the silhouette of the workpiece, the 'Keyence' measurement .This takes only a couple of seconds, but the results are not always accurate. Therefore, the workpieces are transferred to a second measuring machine, called 'MicroVu', that can measure more quality-relevant features, e,g., surface quality and flatness, resulting in more precise results. However, this step takes a couple of minutes for each workpiece. Therefore, a goal is to focus on the first decision point and filter most 'NOK' workpieces using the first measuring step. The first measuring data for two instances can be seen in Fig. 7. The diameter value is shown as a time series of values. The first part where the diameter is between 15-20mm is the thicker part of the valve lifter, whereas the second part where the measurements are around 5-10mm is the thinner end. The highest values at the end, technically do not belong to the workpiece, but the robot gripper that holds the workpiece through the measurement process. The dataset consists of 88 workpieces (36 - 'NOK', 52 - 'OK'). The baseline approach yields the following decision rule:

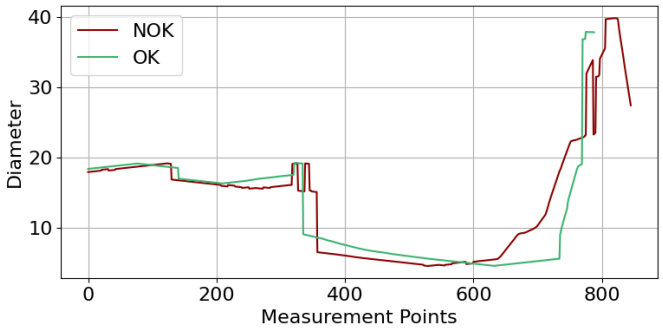

Fig. 7. Manufacturing dataset: visualisation of two instances.

Manufacturing, Baseline:
IF casename <= 2242.5 AND casename <= 2179 AND casename <= 1932.5 AND data_diameterlast <= 27.25 THEN class NOK
Accuracy: 45%
Precision: NOK - 16%, OK - 80%

The resulting decision tree contains 17 nodes, only one applicable rule is shown here, however it is representative of the other rules. We can see that in addition to the last measurement value, the 'casename', which just refers to the instance identifier was used. However, this should not have any impact on the actual quality of the workpiece, which also reflects in relatively low accuracy and precision values.

Manufacturing, EDT-TS - Combined
IF data_diameterlist2_percentchange > 0.16 THEN class NOK
Accuracy: 91%
Precision: NOK - 100%, OK - 90%

The EDT-TS generated rule only has one interval-based condition that refers to the percentage change in the second interval and achieves an accuracy of 91%, as well as precision values of 100% in the 'NOK' class and 91% in the 'OK' class. A high precision value for the 'NOK' class is especially important for this use case, as we want to filter out all 'NOK' pieces beforehand, without unintentionally discarding 'OK' workpieces. The 100% precision value here means that only 'NOK' workpieces are actually classified as 'NOK'.

5 Discussion

Prerequisites for our approach are the existence of a decision point and the availability of process data up to this decision point. Therefore we do not need a complete process model to extract decision rules. The evaluation shows that EDT-TS is feasible and achieves high quality results in terms of accuracy and precision for the used datasets. The approach extracts time series based decision rules and provides them in textual form to the user. For the running example, where the actual decision rule is known, EDT-TS discovers the rule. For the manufacturing dataset the underlying rule is not known, but there are indications that the decision might depend on chips on the workpieces, influencing the quality of the produced parts. Especially the high precision for the 'NOK' is important, as the goal is to discover as many 'NOK' workpieces as possible, without discarding parts that are actually 'OK'. EDT-TS is able to include multiple independent time series and additional numeric data attributes. In terms of generalisability and comprehensiveness, the approach should be able to handle different decision rules that contain a variety of the mentioned decision patterns.

Patterns and use cases for the mentioned patterns might exist that are not covered by the proposed approach yet. Additional feature generation techniques, e.g., additional pattern-based techniques, can be added accordingly.

Limitations and threats to validity: We assume that the measurements are part of an explicit task and can therefore be extracted from the event logs and assigned to the correct process instance. Currently time series with separable effects are taken into account, working with intermingled effects adds another level of complexity as there is no explicit decision point and time series length may vary. So far, the robustness of the approach against noise in the data as well as the computational complexity was not taken into account. Future work will address these limitations. Additionally, future work will focus on including relational rules, e.g. the threshold of a time series is not a fixed value but a variable. Another general limitation is the availability of data. However, as described in Sect. 1 multiple use cases exist, where time series data is logged, but not explicitly part of the event log or not used as time series data.

6 Related Work

Decision mining was coined by [19], describing how to discover decision points in processes. [15] gives an overview of the state-of-the-art, e.g., [13] for the basic approach, [16] for detecting overlapping rules, and [14] for discovering the relationship between variables. These approaches have not considered time series data for decision mining yet. In general, for process mining [22] differentiates between top-down approaches where process mining is applied on process log data augmented with event data, and bottom-up approaches that apply process mining to event data in combination with complex event processing techniques. We can classify our previous work [6, 23, 7] into the top-down category. [6] focuses on how to merge time series data with event logs. The approach relies on manually adding features that seem relevant based on visualisations of the time series. By contrast, EDT-TS specifies textual decision rules instead of clustering instances or calculating the impact of a specific value on the end result. [23] uses time series data, in form of sensor data, to detect concept drifts during runtime, using Dynamic Time Warping and clustering. [7] uses time series data in form of sensor data to predict the process outcome, using manually added, global features as input for the proposed algorithms. Also other recent approaches consider "exogenous data" in combination with process mining. [4] propose an approach to slice time series data and adding the resulting values to specific events as data elements, as well as transformations of this data using global features which is done manually. Bottom-up approaches are proposed by, e.g., [21, 10]. The goal here is to discover or enhance process models based on sensor data. Such approaches can be seen as complementary to EDT-TS.

7 Conclusion

EDT-TS is an approach to discover time series dependent decision rules. Common time series patterns as well as classification algorithms for time series data are analyzed. The analysis results in the definition of three time series dependent decision rule patterns: global feature-based decision rules, interval-based decision rules and pattern-based decision rules. EDT-TS includes three stages. Firstly, the dataset has to be preprocessed, potential time series data elements are found and added as candidate elements. The second phase, generates additional features from these candidate elements, including global features that summarize the entire time series, interval-based features that calculate features for subsequences of the time series and pattern-based features that include the distribution of values in a time series. Decision rules are then built using the generated features and a decision tree. Lastly, the rules are transformed to textual form. The evaluation on two datasets shows high accuracy (91% and 100%) and high precision (between 90% and 100%) values. In future work, we want to address the limitations discussed in Sect. 5, with a special focus on including intermingled effects and multivariate time series as well as making the approach more robust against noise. In addition, runtime detection of decision rules including decision rules based on time series, with changing rules and exceptions, will be part of future work.

Acknowledgements This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via the Austrian Competence Center for Digital Production (CDP) under the contract number 881843.

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Berlin Heidelberg (2016)
2. Abanda, A., Mori, U., Lozano, J.A.: A review on distance based time series classification. Data Min Knowl Disc **33**(2), 378–412 (2019)
3. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Disc **31**(3), 606–660 (2017)
4. Banham, A., Wynn, M.T.: xPM: A Framework for Process Mining with Exogenous Data. In: ICPM Workshops (2021)
5. Dees, M., Hompes, B., van der Aalst, W.M.P.: Events put into context (EPiC). In: International Conference on Process Mining. pp. 65–72 (2020)
6. Dunkl, R., Rinderle-Ma, S., Grossmann, W., Fröschl, K.A.: A Method for Analyzing Time Series Data in Process Mining: Application and Extension of Decision Point Analysis. In: Information Systems Engineering in Complex Environments. pp. 68–84 (2015)
7. Ehrendorfer, M., Mangler, J., Rinderle-Ma, S.: Assessing the impact of context data on process outcomes during runtime. In: Service-Oriented Computing. pp. 3–18. Springer (2021)

8. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Min Knowl Disc **33**(4), 917–963 (2019)
9. Fulcher, B.D.: Feature-based time-series analysis. arXiv:1709.08055 [cs] (Oct 2017), http://arxiv.org/abs/1709.08055, arXiv: 1709.08055
10. Kammerer, K., Pryss, R., Hoppenstedt, B., Sommer, K., Reichert, M.: Process-driven and flow-based processing of industrial sensor data. Sensors **20**(18), 5245 (2020)
11. Kitagawa, G.: Introduction to Time Series Modeling. CRC Press (Apr 2010)
12. Leewis, S., Berkhout, M., Smit, K.: Future Challenges in Decision Mining at Governmental Institutions. In: AMCIS 2020 Proceedings. p. 12 (2020)
13. de Leoni, M., van der Aalst, W.M.P.: Data-aware process mining: discovering decisions in processes using alignments. In: ACM Symposium on Applied Computing. p. 1454 (2013)
14. de Leoni, M., Dumas, M., García-Bañuelos, L.: Discovering Branching Conditions from Business Process Execution Logs. In: Fundamental Approaches to Software Engineering. pp. 114–129 (2013)
15. de Leoni, M., Mannhardt, F.: Decision Discovery in Business Processes. In: Encyclopedia of Big Data Technologies. pp. 1–12 (2018)
16. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Decision Mining Revisited - Discovering Overlapping Rules. In: Advanced Information Systems Engineering. pp. 377–392 (2016)
17. Montgomery, D.C., Jennings, C.L., Kulahci, M.: Introduction to Time Series Analysis and Forecasting. John Wiley & Sons (Apr 2015), google-Books-ID: Xeh8CAAAQBAJ
18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12**(85), 2825–2830 (2011)
19. Rozinat, A., van der Aalst, W.M.P.: Decision Mining in ProM. In: Business Process Management. pp. 420–425 (2006)
20. Scheibel, B., Rinderle-Ma, S.: Comparing decision mining approaches with regard to the meaningfulness of their results. arXiv:2109.07335 [cs] (Sep 2021), http://arxiv.org/abs/2109.07335, arXiv: 2109.07335
21. Seiger, R., Zerbato, F., Burattin, A., García-Bañuelos, L., Weber, B.: Towards iot-driven process event log generation for conformance checking in smart factories. In: EDOC Workshops. pp. 20–26. IEEE (2020)
22. Soffer, P., Hinze, A., Koschmider, A., Ziekow, H., Di Ciccio, C., Koldehofe, B., Kopp, O., Jacobsen, H., Sürmeli, J., Song, W.: From event streams to process models and back: Challenges and opportunities. Inf. Syst. **81**, 181–200 (2019)
23. Stertz, F., Rinderle-Ma, S., Mangler, J.: Analyzing Process Concept Drifts Based on Sensor Event Streams During Runtime. In: Business Process Management. pp. 202–219 (2020)