Enhanced SHAP Calculations for Sequential and Sparse Data in Predictive Process Monitoring

Niyang Bai
niyang.bai@fau.de

May 3, 2025

Abstract

This thesis proposes enhanced SHAP (Shapley Additive Explanations) methods tailored to the needs of Predictive Process Management (PPM), where interpretability is essential, and data often include sequential and sparse structures. For sequential data, the new approaches capture temporal dependencies, enabling a clearer understanding of how past events shape process predictions. For sparse data, optimized SHAP methods reduce dimensionality and computational demands while preserving interpretability. These advancements provide actionable insights into feature importance, supporting real-time, data-driven decisions in PPM and contributing to the broader field of explainable AI by addressing domain-specific challenges in complex, process-based environments. [1]

1 Introduction

Machine learning is increasingly applied in Predictive Process Management (PPM) to predict outcomes within complex business processes. For effective decision-making, these predictions must be transparent and interpretable. SHAP (Shapley Additive Explanations) is widely used for interpretability, attributing model predictions to individual features. However, traditional SHAP methods, like KernelSHAP, encounter challenges in handling high-dimensional or sequential data. In models like LSTMs, SHAP fails to capture temporal dependencies, limiting interpretability. Similarly, in sparse, one-hot encoded datasets, SHAP computations become resource-intensive.

This research aims to address these challenges by developing improved SHAP algorithms tailored to sequential and sparse data models, enhancing interpretability and computational efficiency in PPM applications. Through these improvements, this work enables actionable, transparent insights for predictive management in dynamic, data-intensive settings.

[1] The code for this thesis is available at: github.com/niyangbai/enhanced_shap.git

2 Literature Review

SHAP (Shapley Additive Explanations) has established itself as a critical tool in Explainable AI (XAI) by providing a game-theoretic framework for attributing predictions to features. Lundberg and Lee (2017) introduced SHAP in their seminal paper, laying the foundation for a unified approach to model interpretability. While effective, KernelSHAP—an accessible, model-agnostic SHAP method—suffers from computational inefficiencies, particularly in high-dimensional and sparse data settings (Lundberg and Lee, 2017).

In response to these limitations, Lundberg et al. (2020) proposed TreeSHAP, optimizing SHAP calculations specifically for tree-based models like XGBoost by exploiting the hierarchical structure of trees. This adaptation drastically reduces computational demands, although it is constrained to tree models (Lundberg et al., 2020). Meanwhile, gradient-based SHAP approximations, such as DeepSHAP, have been developed to handle complex models. Inspired by gradient attribution methods (Ancona et al. (2017); Sundararajan et al. (2017)), DeepSHAP computes SHAP values for deep learning models like LSTMs but does not fully account for the temporal dependencies critical to sequential models.

Handling sparse data remains another challenge in SHAP calculations. Traditional methods face high computational costs in these settings, especially with one-hot encoded features. Techniques like Principal Component Analysis (PCA), explored by Jolliffe (2002) and Ng (2004), demonstrate that dimensionality reduction can retain crucial data characteristics while improving computational feasibility. Similarly, Murphy (2012) emphasizes the role of feature selection in managing computational overhead in sparse data, laying a theoretical foundation for efficient SHAP computation in high-dimensional spaces.

In Predictive Process Management (PPM), explainability is essential as organizations increasingly depend on predictive models to inform decisions within complex, dynamic workflows. Early work by der Aalst and Mining (2011) on process mining highlighted the importance of extracting actionable insights from event logs to understand and optimize business processes. More recent studies by De Leoni et al. (2016) and Di Francescomarino et al. (2018) emphasize predictive monitoring's role in anticipating process outcomes, underscoring the need for interpretable models in operational settings. Despite these advances, existing SHAP methodologies lack adaptability for PPM's unique requirements, where data is frequently sequential or sparse. This presents additional interpretability challenges, as conventional SHAP methods are not inherently designed to capture temporal dependencies or manage high-dimensional, sparse data effectively in process-centric applications.

3 Problem Statement

SHAP (Shapley Additive Explanations) values have become essential for interpreting machine learning predictions by attributing the output of a model to

individual features. However, existing SHAP methods face major limitations when applied to high-dimensional or sequential data, both of which are prevalent in Predictive Process Management (PPM). These challenges are particularly evident in two areas:

Interpretability in Sequential Data Models (e.g., LSTM, Transformer): Sequential models, especially those used for time-series data, rely on temporal dependencies, where the impact of each time step on future steps is critical to model predictions. Standard SHAP methods, including gradient-based approaches like DeepSHAP, do not effectively capture these time-dependent relationships, often leading to explanations that fail to reflect the model's reliance on past events. In PPM, where predictions depend heavily on time-ordered process events, lacking this insight into temporal dependencies limits the model's usefulness for real-world decision-making. For stakeholders to take actionable steps based on model outputs, understanding the influence of each process stage in the sequence is essential.

Computational Complexity in Sparse Data for Tabular Models (e.g., XGBoost): Sparse datasets, such as those with numerous one-hot encoded categorical features, increase the dimensionality of the input space, which in turn magnifies the computational demands of SHAP calculations. In such high-dimensional settings, KernelSHAP's combinatorial approach, which evaluates all possible feature subsets, becomes computationally prohibitive. This issue is particularly pressing in PPM, where categorical features often expand the feature space. High computational demands make it difficult to produce timely and efficient interpretations, creating a bottleneck that restricts SHAP's practicality in many real-time PPM applications.

This research aims to address these limitations by developing tailored SHAP algorithms for sequential and sparse data, enhancing both the interpretability and efficiency of SHAP calculations. These innovations will provide more accurate and computationally feasible insights into feature importance, supporting effective decision-making in predictive process management where interpretability is critical for actionable insights.

4 Methodology

The methodology of this research is structured around addressing the limitations of traditional SHAP methods and introducing targeted improvements for sequential and sparse data. The following outlines the benchmark method, as well as the proposed approaches for both primary and secondary objectives.

The naive SHAP method, based on the Shapley value concept from cooperative game theory, provides a comprehensive baseline for explaining a model's predictions. This approach calculates each feature's contribution by considering all possible feature subsets, ensuring consistent and locally accurate explanations. For a model prediction f, the SHAP value for each feature i is calculated as:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[f(S \cup \{i\}) - f(S) \right] \tag{1}$$

where S is a subset of features excluding i, and N is the set of all features. Despite its effectiveness, naive SHAP has an exponential computational complexity of $O(2^n)$, making it infeasible for high-dimensional or sequential data. This limitation provides the impetus for developing more efficient approaches.

The calculation step can be show as follows:

Algorithm 1 Naive SHAP

1: **Input:** Set of all features $X = \{x_1, x_2, \ldots, x_n\}$
2: **for** each feature x_i **do**
3: **for** each subset $S \subseteq N \setminus \{i\}$ **do**
4: Compute the marginal contribution $v(S \cup \{i\}) - v(S)$
5: Weight the marginal contribution by $\frac{|S|!(|N|-|S|-1)!}{|N|!}$
6: **end for**
7: Aggregate the weighted marginal contributions to obtain ϕ_i
8: **end for**
9: **Output:** SHAP values ϕ_i for each feature x_i

Complexity: The computational complexity of naive SHAP is $O(2^n)$ since it requires evaluating the model on every possible subset of n features.

4.1 Primary Objective: Sequential Data Models

4.1.1 Gradient-Based Calculation

GradientSHAP is a method that combines SHAP values with Integrated Gradients to calculate feature importance in deep learning models. This method is particularly useful for sequential models, such as LSTMs, where gradients can approximate the contribution of each time step to the model's output. GradientSHAP calculates SHAP values by integrating gradients along a path from a baseline to the input, providing a smooth estimate of each time step's importance.

Mathematical Proof and Justification:

GradientSHAP approximates each feature's SHAP value by integrating gradients along a path from a baseline X_{ref} to the actual input sequence X. For each time step t in sequence $X = \{x_1, x_2, \ldots, x_T\}$, the SHAP value ϕ_t is given by:

$$\phi_t = \int_{\alpha=0}^{1} (x_t - x_{\text{ref},t}) \cdot \nabla_{x_t} f(X_{\text{ref}} + \alpha(X - X_{\text{ref}}))\, d\alpha$$

where $\nabla_{x_t} f$ represents the gradient of the model's output with respect to x_t, and α is the interpolation parameter that moves from the baseline X_{ref} to

Algorithm 2 GradientSHAP Calculation for Sequential Data

Require: Input sequence $X = \{x_1, x_2, \ldots, x_T\}$, trained LSTM model f, baseline sequence X_{ref}, number of integration steps L

Ensure: SHAP values $\{\phi_t\}_{t=1}^{T}$ for each time step

for $t = 1$ to T **do**
  Initialize SHAP value $\phi_t \leftarrow 0$
  for $l = 1$ to L **do**
    Compute interpolated input $X_l = X_{\text{ref}} + \frac{l}{L} \times (X - X_{\text{ref}})$
    Calculate the gradient $\nabla_{x_t} f(X_l)$ with respect to x_t
    Accumulate $\phi_t \leftarrow \phi_t + \frac{1}{L} \times (x_t - x_{\text{ref},t}) \times \nabla_{x_t} f(X_l)$
  end for
end for
return SHAP values $\{\phi_t\}_{t=1}^{T}$

the input X. This integral calculates the cumulative effect of x_t on the output as the input transitions from the baseline, approximating the SHAP value by summing the gradients along this path.

GradientSHAP's unbiasedness stems from its reliance on the integral, which ensures that the contributions of each time step are computed fairly relative to the baseline. The additivity property is preserved by calculating each time step's impact as a product of its deviation from the baseline and the gradient:

$$f(X) \approx \sum_{t=1}^{T} \phi_t$$

By using integration, GradientSHAP produces a smooth, unbiased estimate of feature importance, with each SHAP value representing an accurate measure of the impact of each time step on the model output.

Complexity:

The complexity of GradientSHAP is driven by the number of integration steps L and the sequence length T. For each time step t, the gradient is calculated L times along the integration path. Therefore, the overall complexity is $O(L \times T)$.

This complexity makes GradientSHAP more computationally efficient than traditional SHAP methods for sequential data, especially when L is relatively small, allowing it to scale well for long sequences in time-series applications.

4.1.2 Recursive Dependency Calculation

RecurrentSHAP is designed to address the limitations of traditional SHAP methods when applied to sequential models like LSTMs, which rely on recursive connections between time steps. Standard SHAP calculations do not capture these temporal dependencies, resulting in explanations that fail to reflect how past events influence future predictions. By incorporating both cell and hidden states of LSTMs into the SHAP calculations, RecurrentSHAP recursively tracks

each time step's influence on subsequent steps, providing an interpretation that aligns with the model's inherent structure.

Algorithm 3 RecurrentSHAP Calculation for Sequential Data

Require: Input sequence $X = \{x_1, x_2, \ldots, x_T\}$, trained LSTM model f, initial cell and hidden states (C_0, h_0)
Ensure: SHAP values $\{\phi_t\}_{t=1}^{T}$ for each time step
Initialize SHAP values: $\phi_t \leftarrow 0$ for $t = 1, \ldots, T$
for $t = 1$ to T **do**
    Calculate the model's output with x_t included, updating hidden state h_t and cell state C_t
    for $j = t$ to T **do**
        Compute the contribution of x_t at time j as $\alpha_j \times (h_j - h_{j-1})$
        Accumulate this contribution to ϕ_t
    end for
end for
return SHAP values $\{\phi_t\}_{t=1}^{T}$

Mathematical Proof and Justification:

To demonstrate that RecurrentSHAP provides an unbiased estimate of feature importance, we show that the contributions of each feature (time step) are computed fairly relative to their influence in the model. For a sequence $X = \{x_1, x_2, \ldots, x_T\}$, the goal is to calculate the SHAP value ϕ_t for each time step t by accounting for its cumulative impact on the model's output through the hidden states at all subsequent steps.

Let the SHAP value for a feature at time step t be given by

$$\phi_t = \sum_{j=t}^{T} \alpha_j \times (h_j - h_{j-1})$$

where α_j is a weight representing the contribution of x_t at time step j, and h_j is the hidden state at time j.

The recursive nature of the LSTM implies that each time step t contributes to all future hidden states h_j for $j \geq t$. Since h_j depends not only on x_j but also on all prior inputs $\{x_1, \ldots, x_{j-1}\}$, this formulation ensures that the SHAP value for x_t accounts for its influence across all dependent states.

We validate that RecurrentSHAP is unbiased by ensuring it satisfies the additivity property. Suppose that each time step's contribution to the model output $f(X)$ is decomposed as:

$$f(X) = \sum_{t=1}^{T} \phi_t$$

This summation holds because each term ϕ_t represents the marginal contribution of x_t when aggregated over all future time steps. By calculating the

SHAP values recursively, RecurrentSHAP respects the dependency structure of the LSTM, ensuring that contributions are distributed fairly without over- or under-emphasizing any particular time step. This recursive summation guarantees that the sum of SHAP values equals the model's output, demonstrating an unbiased and consistent distribution of feature importance across the sequence.

Complexity:

The complexity of RecurrentSHAP is driven by the nested loop structure in the algorithm. For each time step t in the input sequence of length T, we calculate the contribution of x_t to every subsequent time step $j \geq t$. This requires evaluating approximately $T(T+1)/2$ operations across the sequence.

Thus, the time complexity of RecurrentSHAP is $O(T^2)$. This quadratic complexity represents a significant improvement over traditional SHAP calculations, which have exponential complexity, and makes RecurrentSHAP feasible for reasonably long sequences in sequential models like LSTMs.

4.1.3 Calculation with Attention Mechanisms

Attention-SHAP is designed to leverage attention mechanisms within LSTM models to address the challenge of accurately attributing feature importance across time steps. In models equipped with an attention layer, each time step is assigned a relevance weight based on its influence on the overall prediction. By incorporating these attention weights into SHAP calculations, Attention-SHAP produces feature attributions that better reflect the model's focus across the sequence, enhancing interpretability, especially in applications where specific events are more influential than others.

Algorithm 4 Attention-SHAP Calculation for Sequential Data

Require: Input sequence $X = \{x_1, x_2, \ldots, x_T\}$, trained LSTM model f with attention weights $\{a_t\}_{t=1}^{T}$
Ensure: SHAP values $\{\phi_t\}_{t=1}^{T}$ for each time step
Initialize SHAP values: $\phi_t \leftarrow 0$ for $t = 1, \ldots, T$
for $t = 1$ to T **do**
    Calculate the model's output with x_t included, generating attention weight a_t
    for $j = t$ to T **do**
        Compute the weighted contribution of x_t at time j as $a_t \times (h_j - h_{j-1})$
        Accumulate this contribution to ϕ_t
    end for
end for
return SHAP values $\{\phi_t\}_{t=1}^{T}$

Mathematical Proof and Justification:

Attention-SHAP aims to provide an unbiased and consistent feature attribution by weighting each time step's SHAP value according to its attention-derived relevance. Given a sequence $X = \{x_1, x_2, \ldots, x_T\}$ and associated attention weights $\{a_t\}_{t=1}^{T}$, the SHAP value ϕ_t for each time step t is calculated as:

$$\phi_t = a_t \times \sum_{j=t}^{T} (h_j - h_{j-1})$$

Here, a_t is the attention weight assigned to time step t, and h_j denotes the hidden state at time step j. By incorporating attention weights, Attention-SHAP scales each SHAP value based on the LSTM's internal focus, ensuring that each time step's SHAP value reflects its relative importance in the model's attention mechanism.

To demonstrate that this approach remains unbiased, we verify that Attention-SHAP satisfies the additivity property. This property ensures that the total model output $f(X)$ can be decomposed as the sum of the SHAP values for each time step:

$$f(X) = \sum_{t=1}^{T} \phi_t$$

In Attention-SHAP, each term ϕ_t represents a weighted marginal contribution that respects the attention structure in the model. The attention weights a_t ensure that each time step's influence is proportionally represented, preserving the overall consistency of the SHAP values with the model's output. Consequently, Attention-SHAP maintains the unbiased distribution of feature importance across the sequence, accounting for the model's selective focus on different time steps without introducing bias.

Complexity:

The complexity of Attention-SHAP arises from the nested loop structure, where we compute the SHAP contribution of each time step t for every subsequent time step $j \geq t$. Given a sequence length T, this requires approximately $T(T+1)/2$ operations.

Thus, the time complexity of Attention-SHAP is $O(T^2)$, similar to RecurrentSHAP. The quadratic complexity allows Attention-SHAP to remain computationally feasible for reasonably long sequences, while incorporating additional insights from attention weights to improve interpretability in LSTM models.

4.1.4 Sequential Perturbation SHAP

Sequential Perturbation SHAP (SP-SHAP) is a method developed to estimate SHAP values for sequential models by perturbing individual time steps and measuring their impact on the model's output. Unlike traditional SHAP methods, which calculate feature attributions by evaluating all feature subsets, SP-SHAP isolates each time step's contribution by observing changes in the prediction when each time step is perturbed. This approach is particularly effective in time-series models, as it allows for efficient evaluation of each event's influence on the sequence without extensive combinatorial computations.

Mathematical Proof and Justification:

Algorithm 5 SP-SHAP Calculation for Sequential Data

Require: Input sequence $X = \{x_1, x_2, \ldots, x_T\}$, trained LSTM model f, perturbation function δ

Ensure: SHAP values $\{\phi_t\}_{t=1}^{T}$ for each time step

1: Initialize SHAP values: $\phi_t \leftarrow 0$ for $t = 1, \ldots, T$
2: **for** $t = 1$ to T **do**
3: Generate perturbed sequence $X'_t = X - \delta(x_t)$ by applying perturbation δ to time step x_t
4: Compute the model outputs for the original sequence $f(X)$ and perturbed sequence $f(X'_t)$
5: Calculate the SHAP value for x_t as $\phi_t = f(X) - f(X'_t)$
6: **end for**
7: **return** SHAP values $\{\phi_t\}_{t=1}^{T}$

SP-SHAP provides an unbiased approximation of each time step's contribution by systematically perturbing individual time steps and observing the resulting change in the model's output. For a sequence $X = \{x_1, x_2, \ldots, x_T\}$, the SHAP value ϕ_t for each time step t is calculated as:

$$\phi_t = f(X) - f(X - \delta(x_t))$$

where $\delta(x_t)$ represents the perturbation applied to x_t, and $X - \delta(x_t)$ denotes the sequence with x_t perturbed. The difference $f(X) - f(X - \delta(x_t))$ captures the marginal contribution of x_t to the overall prediction, approximating the SHAP value without evaluating all feature subsets.

To verify that SP-SHAP is unbiased, we note that the SHAP values satisfy the additivity property, meaning that the sum of all perturbed contributions approximates the model's output $f(X)$. Given that each SHAP value ϕ_t is computed by isolating the effect of perturbing x_t, the total model output can be expressed as the sum of individual contributions:

$$f(X) \approx \sum_{t=1}^{T} \phi_t$$

This summation holds because each ϕ_t represents the impact of x_t in the sequence, and the perturbation-based approach ensures that each time step's contribution is independently assessed. Thus, SP-SHAP maintains an unbiased approximation of the model's output while simplifying the computational process by focusing on individual perturbations rather than exhaustive subset evaluations.

Complexity:

The complexity of SP-SHAP depends on the number of perturbations applied to the sequence. For a sequence length T, each time step is perturbed once, and the model is evaluated twice per time step (for the original and perturbed sequences). Therefore, SP-SHAP has a linear complexity of $O(T)$.

This linear complexity makes SP-SHAP highly efficient for long sequences, allowing it to scale well with sequence length and providing a feasible alternative to traditional SHAP methods for sequential data.

4.1.5 Time-Series Based Calculation

TimeSHAP is an adaptation of SHAP tailored specifically for time-series data, which aims to capture the unique temporal dependencies of sequential models like LSTM. In TimeSHAP, each time step's SHAP value is computed in a way that respects the temporal ordering of events, ensuring that the impact of each time step is assessed sequentially. This approach provides clear interpretability in time-sensitive applications by accounting for how past events influence future predictions within the model.

Algorithm 6 TimeSHAP Calculation for Sequential Data

Require: Input sequence $X = \{x_1, x_2, \ldots, x_T\}$, trained LSTM model f, background set of reference sequences $\{X_b\}$
Ensure: SHAP values $\{\phi_t\}_{t=1}^{T}$ for each time step
1: **for** $t = 1$ to T **do**
2: **for** each background sequence X_b **do**
3: Construct masked sequence $X_t' = \text{mask}(X, t)$ by removing x_t
4: Compute model outputs for X and X_t' as $f(X)$ and $f(X_t')$, respectively
5: Calculate the contribution of x_t as $f(X) - f(X_t')$
6: **end for**
7: Average contributions across all background sequences to obtain ϕ_t
8: **end for**
9: **return** SHAP values $\{\phi_t\}_{t=1}^{T}$

Mathematical Proof and Justification:

TimeSHAP assigns an unbiased estimate of feature importance by calculating the marginal contribution of each time step relative to a baseline. For each time step t in sequence $X = \{x_1, x_2, \ldots, x_T\}$, TimeSHAP calculates the SHAP value ϕ_t as the difference between the model's output with and without x_t, averaged over a background set $\{X_b\}$ of reference sequences. The SHAP value ϕ_t is given by:

$$\phi_t = \mathbb{E}_{X_b}\left[f(X) - f(\text{mask}(X, t))\right]$$

where $\text{mask}(X, t)$ denotes the sequence with x_t removed, and $\mathbb{E}_{X_b}$ represents the expectation over the background set $\{X_b\}$. By averaging contributions across different reference sequences, TimeSHAP captures the marginal impact of x_t relative to other potential values, aligning with the principles of Shapley values.

This approach satisfies the additivity property, as the sum of the SHAP values approximates the model's output $f(X)$:

$$f(X) \approx \sum_{t=1}^{T} \phi_t$$

TimeSHAP maintains this unbiasedness by evaluating each time step relative to the baseline, ensuring that the calculated SHAP values collectively approximate the model output, with each time step's impact assessed independently.

Complexity:

The complexity of TimeSHAP is influenced by the number of background sequences B and the length T of the input sequence. For each time step, the model is evaluated with the original and masked sequences for each background instance, leading to a total complexity of $O(B \times T)$.

This linear complexity with respect to both the sequence length and the number of background samples allows TimeSHAP to scale efficiently for sequential data, making it feasible for time-series applications where interpretability is essential.

4.2 Secondary Objective: Sparse Data Models

4.2.1 Stochastic Group and KernelSHAP - Selecting Only Non-Zeros

This algorithm aims to simplify the SHAP value computation for sparse data by focusing only on the non-zero features. Since zero entries in one-hot encoded data do not contribute to feature interactions, excluding them from the SHAP computation reduces computational complexity without affecting the results.

Algorithm 7 Stochastic Group and KernelSHAP (Selecting Non-Zeros)

Input: Set of features $X = \{x_1, x_2, \ldots, x_n\}$
Identify non-zero one-hot encoded features $Z \subseteq X$
for each feature $x_i \in Z$ **do**
    for each subset $S \subseteq Z \setminus \{i\}$ **do**
        Compute the marginal contribution $v(S \cup \{i\}) - v(S)$
        Weight the marginal contribution by $\frac{|S|!(|Z|-|S|-1)!}{|Z|!}$
    end for
    Aggregate the weighted marginal contributions to obtain ϕ_i
end for
Aggregate the SHAP values over multiple stochastic selections
Output: SHAP values ϕ_i for each feature x_i

Mathematical Proof and Justification:

Consider the set of features $X = \{x_1, x_2, \ldots, x_n\}$ where some features are one-hot encoded and hence sparse. Let $Z \subseteq X$ be the set of non-zero features, and let Z_0 be the set of features that are zero in the instance being explained.

For KernelSHAP, we select a subset S of Z and compute the SHAP value as shown in Equation 1.

Since zeros in one-hot encoded features indicate the absence of categorical attributes, their contribution to the value function $v(S)$ is effectively neutral. Specifically, for any $S \subseteq Z$:

$$v(S \cup Z_0) = v(S)$$

This implies that the value of the function does not change with the inclusion of zero-valued features, meaning their marginal contribution is zero:

$$v(S \cup \{i\}) - v(S) = 0 \quad \text{if} \quad i \in Z_0$$

Therefore, the SHAP value for a zero feature $i \in Z_0$ is:

$$\begin{aligned}
\phi_i &= \sum_{S \subseteq Z \setminus \{i\}} \frac{|S|!(|Z| - |S| - 1)!}{|Z|!} \cdot (v(S \cup \{i\}) - v(S)) \\
&= \sum_{S \subseteq Z \setminus \{i\}} \frac{|S|!(|Z| - |S| - 1)!}{|Z|!} \cdot 0 \\
&= 0
\end{aligned}$$

Thus, zero-valued features do not contribute to the SHAP value, confirming that they are not relevant in the computation. The expected value of the SHAP value remains the same whether zero features are included or excluded:

$$\begin{aligned}
\mathbb{E}[\phi_i^{\text{Stochastic}}] &= \mathbb{E}\left[\sum_{S \subseteq Z \setminus \{i\}} \frac{|S|!(|Z| - |S| - 1)!}{|Z|!}(v(S \cup \{i\}) - v(S))\right] \\
&= \phi_i
\end{aligned}$$

Hence, the expected value of the SHAP value using stochastic selection of non-zero features is equivalent to the naive SHAP value.

Complexity: Let m be the number of non-zero features in a particular instance. The computational complexity of this method is $O(2^m)$, which is significantly lower than $O(2^n)$ if $m \ll n$.

4.2.2 PCA, Stochastic Grouping, and KernelSHAP

This algorithm integrates Principal Component Analysis (PCA) to reduce the dimensionality of the original features while retaining the most significant components. The reduced feature set, combined with the non-zero one-hot encoded features, allows for efficient SHAP value computation. The results are then mapped back to the original feature space.

Algorithm 8 PCA, Stochastic Grouping, and KernelSHAP

Input: Set of original features X
Apply PCA to the original features and select the top k principal components
Combine the selected principal components with non-zero one-hot encoded features Z
for each feature $x_i \in X' = \{\text{top } k \text{ principal components}\} \cup Z$ **do**
    for each subset $S' \subseteq X' \setminus \{i\}$ **do**
        Compute the marginal contribution $v(S' \cup \{i\}) - v(S')$
        Weight the marginal contribution by $\frac{|S'|!(|X'|-|S'|-1)!}{|X'|!}$
    end for
    Aggregate the weighted marginal contributions to obtain ϕ'_i
end for
Map the SHAP values back to the original feature space
Output: SHAP values ϕ_i for each feature x_i

Mathematical Proof and Justification:

Let X be the original feature set and X' be the transformed feature set after applying PCA, where X' consists of the top k principal components.

The SHAP value is computed on X':

$$\phi'_i = \sum_{S' \subseteq X' \setminus \{i'\}} \frac{|S'|!(|X'| - |S'| - 1)!}{|X'|!} (v(S' \cup \{i'\}) - v(S'))$$

Mapping back to the original space, the SHAP values ϕ_i can be approximated by:

$$\phi_i \approx \sum_{j=1}^{k} \alpha_{ij} \phi'_j$$

where α_{ij} are the coefficients that map the principal components back to the original features.

The expected value of the SHAP value using PCA and stochastic grouping is:

$$\mathbb{E}[\phi_i^{\text{PCA}}] = \mathbb{E}\left[\sum_{j=1}^{k} \alpha_{ij} \sum_{S' \subseteq X' \setminus \{j\}} \frac{|S'|!(|X'| - |S'| - 1)!}{|X'|!} (v(S' \cup \{j\}) - v(S'))\right]$$

Given the linearity of expectation and the properties of PCA preserving variance and interactions:

$$\mathbb{E}[\phi_i^{\text{PCA}}] = \phi_i$$

Thus, the expected value of the SHAP value using PCA and stochastic grouping is equivalent to the naive SHAP value.

Complexity:

- PCA transformation has a complexity of $O(n^3)$ for computing the principal components.
- After PCA, the complexity of KernelSHAP on k components and m nonzero features is $O(2^k \cdot 2^m)$. If k and m are significantly smaller than n, this complexity is much lower than $O(2^n)$.

4.2.3 Stochastic Grouping and KernelSHAP with Grouped Feature Setup

This algorithm groups one-hot encoded features based on their original categorical attributes, allowing for a more structured approach to feature selection. By applying stochastic grouping within these groups, the algorithm maintains the integrity of feature interactions and reduces dimensionality.

Algorithm 9 Stochastic Grouping and KernelSHAP with Grouped Feature Setup

Input: Set of one-hot encoded features grouped by original categorical attributes $G = \{G_1, G_2, \ldots, G_m\}$
for each group $G_j \in G$ **do**
  for each feature $x_i \in G_j$ **do**
    for each subset $S_j \subseteq G_j \setminus \{i\}$ **do**
      Compute the marginal contribution $v(S_j \cup \{i\}) - v(S_j)$
      Weight the marginal contribution by $\frac{|S_j|!(|G_j|-|S_j|-1)!}{|G_j|!}$
    end for
    Aggregate the weighted marginal contributions to obtain ϕ_{ij}
  end for
end for
Aggregate the SHAP values across all groups to obtain ϕ_i
Output: SHAP values ϕ_i for each feature x_i

Mathematical Proof and Justification:

Let $G = \{G_1, G_2, \ldots, G_m\}$ be the groups of features, where each group G_j contains related one-hot encoded features.

For KernelSHAP with grouped features, we compute the SHAP value within each group G_j:

$$\phi_{ij} = \sum_{S_j \subseteq G_j \setminus \{i\}} \frac{|S_j|!(|G_j| - |S_j| - 1)!}{|G_j|!} (v(S_j \cup \{i\}) - v(S_j))$$

The overall SHAP value for feature x_i considering the grouped setup is:

$$\phi_i = \sum_{j=1}^{m} \phi_{ij}$$

Since the grouping preserves the interaction patterns and contributions of features, the expected value is:

$$\mathbb{E}[\phi_i^{\text{Grouped}}] = \mathbb{E}\left[\sum_{j=1}^{m} \sum_{S_j \subseteq G_j \setminus \{i\}} \frac{|S_j|!(|G_j| - |S_j| - 1)!}{|G_j|!}(v(S_j \cup \{i\}) - v(S_j))\right]$$

Again, by the linearity of expectation and the grouping preserving feature interactions:

$$\mathbb{E}[\phi_i^{\text{Grouped}}] = \phi_i$$

Thus, the expected value of the SHAP value using the grouped feature setup is equivalent to the naive SHAP value.

Complexity:

- Let g be the number of groups and h be the average number of features per group.
- The complexity of KernelSHAP for each group is $O(2^h)$.
- Since there are g groups, the total complexity is $O(g \cdot 2^h)$. If h is significantly smaller than n, this complexity is much lower than $O(2^n)$.

5 Implementation on Predictive Process Management (PPM)

The proposed SHAP algorithms are implemented to improve interpretability and computational efficiency in Predictive Process Management (PPM) applications, where timely and transparent insights are crucial. In PPM, models often rely on sequential data, representing process timelines, or sparse data, capturing categorical attributes that expand feature dimensions.

For the primary objective, which focuses on enhancing SHAP calculations in sequential models, methods like RecurrentSHAP, Attention-SHAP, SP-SHAP, and TimeSHAP provide insights into how each event or step in a process influences predicted outcomes. These methods are tailored to capture temporal dependencies, making them particularly useful for time-sensitive process monitoring. By attributing importance to individual time steps based on their impact on future outcomes, these approaches facilitate actionable insights, allowing process managers to identify critical events within a sequence that may require intervention or optimization.

For the secondary objective, optimized SHAP methods for sparse tabular data improve interpretability in PPM settings where categorical features are common, such as tracking discrete states in business processes. The proposed algorithms, including Stochastic Grouping, PCA with KernelSHAP, and Grouped Feature Setup, efficiently handle high-dimensional data, reducing the computational burden while preserving interpretability. These methods support clear attributions of feature importance even in sparse data settings, providing PPM

stakeholders with insights into which process attributes contribute most significantly to predictions. This capability enhances decision-making by highlighting key process characteristics that may affect process efficiency, compliance, or other performance indicators.

Through these tailored SHAP implementations, both sequential and sparse data challenges in PPM are addressed, enabling more accurate, timely, and interpretable insights into process outcomes.

6 Evaluation Criteria

The proposed SHAP algorithms will be evaluated based on three primary criteria: accuracy, computational efficiency, and interpretability. These metrics assess each method's effectiveness in enhancing SHAP calculations for Predictive Process Management (PPM) and in addressing the challenges posed by sequential and sparse data.

Accuracy The accuracy of each SHAP method will be evaluated by comparing its feature importance values to those of established benchmark methods, such as KernelSHAP, across different PPM datasets. For sequential models, accuracy is measured by the alignment of time-step SHAP values with known or expected influential events in the process sequence. For sparse data, accuracy is assessed by examining how well the SHAP values highlight critical categorical features that are known to affect process outcomes. This criterion ensures that each proposed method provides valid and reliable explanations.

Computational Efficiency Efficiency is evaluated in terms of computational time and resource usage. Each method's performance will be benchmarked against KernelSHAP to measure improvements in scalability and feasibility, particularly for long sequences and high-dimensional sparse data. For sequential data, the efficiency of RecurrentSHAP, Attention-SHAP, SP-SHAP, TimeSHAP, and GradientSHAP will be assessed to ensure that they remain computationally feasible for real-time process monitoring. For sparse data, the performance of Stochastic Grouping, PCA with KernelSHAP, and Grouped Feature Setup will be examined to validate their ability to handle high-dimensional inputs efficiently.

Interpretability Interpretability is assessed based on the clarity and coherence of the SHAP values in conveying feature importance to PPM stakeholders. For sequential data, interpretability is measured by the ease with which stakeholders can understand time-step contributions to process outcomes, enabling actionable insights into critical events. For sparse data, interpretability is evaluated by how clearly the SHAP values indicate important categorical features, allowing stakeholders to identify specific process attributes that drive predictions. This criterion ensures that the proposed SHAP methods produce intuitive and actionable explanations that facilitate effective decision-making in PPM.

Through these evaluation criteria, the effectiveness of each SHAP method in enhancing interpretability, computational efficiency, and accuracy within PPM applications will be rigorously assessed, supporting their adoption in process-based predictive modeling.

7 Expected Outcomes

This research is expected to yield improved SHAP algorithms tailored to the unique demands of Predictive Process Management (PPM), specifically addressing the interpretability and computational challenges associated with sequential and sparse data.

For the primary objective, the proposed methods for sequential data, including RecurrentSHAP, Attention-SHAP, SP-SHAP, TimeSHAP, and GradientSHAP, are anticipated to provide clearer, temporally-aware explanations of model predictions. By capturing temporal dependencies, these methods should allow stakeholders to trace predictions back to critical events within a process sequence, offering actionable insights for process optimization, intervention, and decision-making. Improved accuracy and computational feasibility in handling long sequences are also expected, enabling these methods to support real-time applications in process monitoring.

For the secondary objective, the optimized SHAP algorithms for sparse data, such as Stochastic Grouping, PCA with KernelSHAP, and Grouped Feature Setup, are expected to enhance interpretability without the excessive computational overhead typical in high-dimensional settings. By focusing on non-zero features and reducing feature dimensionality efficiently, these methods are anticipated to offer clear insights into categorical attributes that significantly impact process outcomes. This clarity should help process managers identify key drivers of performance, compliance, or efficiency within large datasets, supporting data-driven decision-making in PPM.

8 Novelty of the Thesis

This thesis introduces novel approaches to SHAP value calculation specifically tailored for sequential and sparse data, filling a significant gap in the field of explainable AI, particularly within the context of Predictive Process Management (PPM).

For sequential data, methods such as RecurrentSHAP, Attention-SHAP, SP-SHAP, TimeSHAP, and GradientSHAP offer the first targeted adaptation of SHAP calculations to capture temporal dependencies in time-series models. These methods extend traditional SHAP frameworks by explicitly accounting for the influence of prior time steps on model predictions, producing temporally-aware explanations that better reflect the dynamics of sequential data. This approach represents a critical innovation in XAI for time-sensitive applications, enabling PPM stakeholders to derive clearer insights into how specific events

influence process outcomes over time.

For sparse data, this thesis introduces optimizations in SHAP computations that effectively manage high-dimensional inputs, such as one-hot encoded categorical variables, which are common in PPM datasets. Techniques like Stochastic Grouping, PCA with KernelSHAP, and Grouped Feature Setup are specifically designed to reduce computational complexity while preserving interpretability in sparse data environments. By focusing on computational efficiency and dimensionality reduction, these methods allow for scalable SHAP calculations in real-time applications, supporting timely and interpretable insights in large-scale PPM deployments.

This research advances SHAP methodologies beyond general-purpose implementations, adapting them to the complex requirements of PPM. By addressing both sequential and sparse data challenges, the thesis provides a pioneering contribution to explainable AI, offering new tools that enhance the interpretability, accuracy, and efficiency of predictive modeling in process-based applications. These contributions are expected to drive the adoption of explainable AI in PPM and inspire further research in XAI tailored for domain-specific needs.

9 Conclusion

This thesis presents a set of novel SHAP algorithms specifically designed to address the interpretability and computational challenges inherent in Predictive Process Management (PPM) applications involving sequential and sparse data. By introducing RecurrentSHAP, Attention-SHAP, SP-SHAP, TimeSHAP, and GradientSHAP for sequential data, along with optimized methods for sparse data such as Stochastic Grouping, PCA with KernelSHAP, and Grouped Feature Setup, this research advances the state of explainable AI by tailoring SHAP calculations to the unique needs of PPM. These contributions enhance model interpretability by capturing temporal dependencies and reducing computational costs, enabling actionable insights into process outcomes. This work not only improves the feasibility of deploying SHAP-based interpretations in real-time PPM but also provides a foundation for further research in explainable AI for domain-specific applications.

References

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. (2017). Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*.

De Leoni, M., Van Der Aalst, W. M., and Dees, M. (2016). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Systems*, 56:235–257.

der Aalst, V. and Mining, W. P. (2011). Discovery, conformance and enhancement of business processes. *Media; Springer: Berlin/Heidelberg, Germany*, 136.

Di Francescomarino, C., Ghidini, C., Maggi, F. M., and Milani, F. (2018). Predictive process monitoring methods: Which one suits me best? In *International conference on business process management*, pages 462–479. Springer.

Jolliffe, I. T. (2002). Principal component analysis and factor analysis. *Principal component analysis*, pages 150–166.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.