National Research University Higher School of Economics

Faculty of Computer Science

Programme: Data Science and Business Analytics

BACHELOR'S THESIS

Research Project

Explainable Predictive Process Monitoring

Prepared by the student of Group **211**, Year 4 (year of study),
Ageev Ruslan

Thesis Supervisor:
Associate Professor, Deputy Dean for Research, **Alexey A. Mitsyuk**

Moscow
2025

Abstract

Keywords: Predictive Process Monitoring, Process Mining, Graph Neural Networks, Explainable AI (XAI), Interpretability, Next-Activity Prediction.

Predictive Process Monitoring (PPM) allows one to estimate the future behavior of business processes in advance and support operational decisions. Graph neural networks (GNNs) demonstrate high predictive accuracy in this task, but their explainability remains critically limited: universal XAI approaches (SHAP, LIME) poorly capture the cyclic and temporal nature of event logs, and existing explainers for GNNs are not suitable for the semantics of edge and node types. This paper presents TAPGExplainer, the first type-sensitive PGExplainer extender specifically adapted to heterogeneous process graphs. The algorithm trains separate masks for each type of the relations, maximizing the mutual information between the explanatory subgraph and the original prediction. An in-depth evaluation of PPM solutions identified PROPHET as the most suitable model. The TAPGExplainer interpreter was built into the PROPHET model and tested on publicly available process logs. A systematic comparative analysis with state-of-the-art explainers (GNNExplainer, SubgraphX, original PGExplainer) was performed, showing a steady improvement in interpretability, compactness, and speed of explanation generation on all datasets. This work bridges the gap between the high accuracy of graph predictors and the requirement for their transparency.

Contents

1 Introduction

Process mining seeks to improve operational processes by analyzing the event data that information systems record at run time [36]. Following the taxonomy of van der Aalst [37], the field comprises four core tasks: *process discovery*, *conformance checking*, *process enhancement*, and *predictive monitoring*. Operational processes are repetitive sequences of activities that occur across domains—from manufacturing and logistics to finance, healthcare, and public administration—to deliver a product or service

Predictive Process Monitoring (PPM) focuses on ongoing cases, with the aim of forecasting their future evolution based on historical event logs. By learning from completed instances, PPM techniques can estimate the next activity, predict the remaining sequence (*suffix*), determine the eventual outcome, or compute the residual duration of a case. Such real-time insights support timely interventions whenever delays, deviations, or failures are imminent.

Driven by the availability of large-scale logs, recent PPM research has embraced machinelearning and deeplearning models. Recurrent neural networks, particularly, LSTM and GRU variants [16,34], constituted an early wave of data-driven approaches. Transformer architectures equipped with self-attention, which is well suited to modeling long-range dependencies in event sequences, have since gained traction [28]. More recently, *graph-based* methods have emerged: By representing each trace prefix as a heterogeneous graph whose nodes and edges encode control flow or data flow relations, *Graph Neural Networks* (GNNs) can exploit rich structural information [9, 27, 29]. Exploratory work has even examined large language models in the context of PPM [26] [40]. Although classical baselines such as transition systems and stochastic Petri nets remain relevant, deep-learning approaches now dominate the literature.

As predictive models grow in complexity and adoption, their *explainability* becomes critical, especially when recommendations guide high-stakes decisions. Generic post hoc tools such as SHAP and LIME are applicable, yet they neither exploit the sequential and temporal characteristics of business processes nor expose the internal reasoning of the model. Empirical studies confirm these limitations: Rizzi *et al.* report explanation faithfulness scores below 0.4 for LIME-like methods in PPM logs [32], while Galanti *et al.* observe low practitioner usability ratings for saliency-based visualizations [14]. Consequently, existing XAI techniques provide limited support for process-aware decision-making.

GNN-based predictors achieve state-of-the-art accuracy by modeling process structure [29]; however, generating *faithful* and *comprehensive* explanations for GNN output remains an open challenge. To address this gap, we introduce a novel explainer, **Type-Aware PGExplainer (TAPGExplainer)**, which leverages the semantic types of nodes and edges present in process graphs. By coupling TAPGExplainer with leading next-event predictors, this study advances the state of *explainable* predictive process monitoring and provides a first rigorous benchmark of explanation quality within PPM.

The remainder of the paper is organised as follows. Section 2 presents background concepts and definitions. Section 3 surveys related work on predictive process monitoring and explainable AI. Section 4 articulates the research gap and objectives. Section 5 details the proposed method, including the graph-encoding scheme and the novel TAPGExplainer. Section 6 describes the datasets and evaluation metrics. Section 7 reports the experimental setup and results. Section 8 discusses the findings and their implications. Section 9 concludes the paper and outlines avenues for future research.

This study emphasises its scientific novelty by introducing the first type-aware explanations for GNN-based PPM and providing a comprehensive empirical evaluation of predictive and explanatory performance

2 Background & Definitions

Predictive Process Monitoring (PPM) builds on the formal concepts of process mining. This section introduces the minimum terminology required for the rest of the thesis and clarifies the prediction tasks addressed.

2.1 Formal Notions

Events. Let $\mathcal{A}$ be the finite set of event attributes (e.g., *activity*, *resource*, *cost*). Each attribute $a \in \mathcal{A}$ has a domain $\mathcal{D}(a)$. Define the universe of attribute values as $\mathcal{D} = \bigcup_{a \in \mathcal{A}} \mathcal{D}(a)$. An *event* is a partial function

$$e : \mathcal{A} \nrightarrow \mathcal{D}, \qquad e(a) \in \mathcal{D}(a),$$

which maps a subset of attributes to concrete values. The attributes *activity* and *timestamp* are mandatory.

Trace. With $\mathcal{E}$ the universe of events, a *trace* is a finite, non-empty sequence $\sigma = \langle e_1, e_2, \ldots, e_n \rangle \in \mathcal{E}^*$ describing one process instance.

Event log. An *event log* is a multiset $L \in \mathcal{M}(\mathcal{E}^*)$ of traces, where $\mathcal{M}(\cdot)$ denotes the bag (multiset) constructor.

Prefix and suffix. For $1 \leq k < n$, the k-length prefix and suffix of a trace are

$$\mathrm{hd}_k(\sigma) = \langle e_1, \ldots, e_k \rangle, \qquad \mathrm{tl}_k(\sigma) = \langle e_{k+1}, \ldots, e_n \rangle.$$

2.2 Notation and Abbreviations

All the Notions and Abbreviations for this work are in the following table 1.

Table 1: List of abbreviations used throughout the thesis

Abbreviation	**Meaning**
BPM	Business Process Management
BPI	Business Process Intelligence (Challenge)
CPU	Central Processing Unit
DL	Deep Learning
GNN	Graph Neural Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
PPM	Predictive Process Monitoring
RNN	Recurrent Neural Network
SUS	System Usability Scale
TAPG	Type-Aware PGExplainer
XAI	Explainable Artificial Intelligence

2.3 Predictive Process Monitoring Task Taxonomy

Following Di Francescomarino and Ghidini [35], PPM approaches can be classified along three orthogonal dimensions (Fig. 1).
Let Π be the set of all prefixes $\mathrm{hd}_k(\sigma)$ extracted from L. For each prediction task we define a mapping Ω:

Definition 1 (Next activity)**.**

$$\Omega_A : \Pi \to \mathcal{D}(\textit{activity}), \quad \Omega_A\big(\mathrm{hd}_k(\sigma)\big) = a',$$

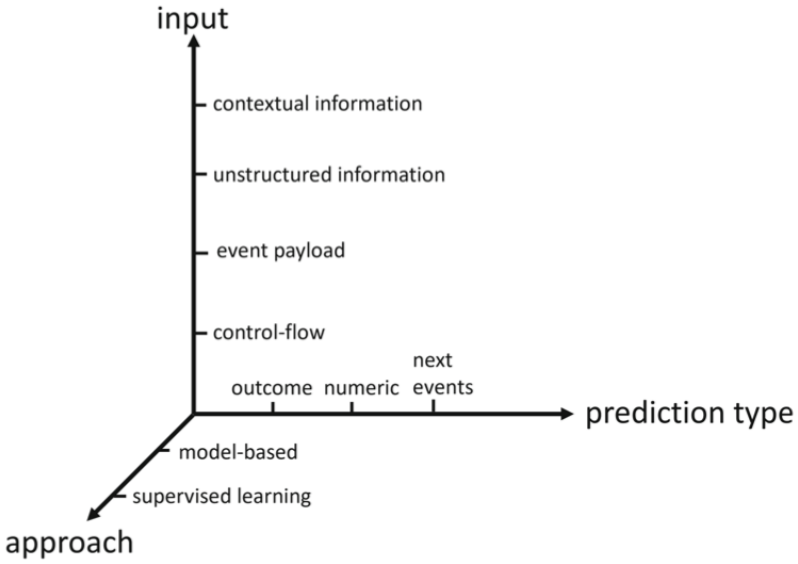


Figure 1: Predictive Process monitoring along 3 dimensions.

where a' denotes the next activity.

Definition 2 (Next attribute)**.** *For $d \in \mathcal{A}$,*

$$\Omega_d\big(\mathrm{hd}_k(\sigma)\big) = v', \quad v' \in \mathcal{D}(d).$$

Definition 3 (Next timestamp)**.**

$$\Omega_T\big(\mathrm{hd}_k(\sigma)\big) = \pi_T(e_k) + t',$$

where π_T extracts timestamps and t' is the predicted inter-event time.

Definition 4 (Outcome)**.**

$$\Omega_O\big(\mathrm{hd}_k(\sigma)\big) = o', \quad o' \in \mathcal{O}.$$

Suffix prediction

Activity or attribute suffixes can be produced recursively by feeding back newly predicted elements until an `[EOC]` marker is, reached or generated in a single step. Remaining-time estimation follows by summing predicted inter-event durations.

Types of approaches

PPM solutions fall into two broad families:

Model-based. The first one is model-based, where an explicit process model is used as the basis for making predictions. This model can be either discovered directly from the event log or can already exist and be enriched using data from the log. During runtime, predictions are made based on this explicit model, which reflects the structure and control flow of the process [**?**]. These methods are transparent but under-exploit fine-grained temporal detail [35].

Machine-learning based. The second category includes approaches that apply machine learning or statistical methods, such as classification, regression, and neural networks. These models work with implicit representations of the process. They extract features from event logs and use them as input for training predictive models. In this case, the structure of the process is not explicitly modeled but instead is learned through data-driven techniques.

Most existing model-based approaches do not fully utilize the structured and temporal information available in event logs. [29] However, these methods are generally more interpretable due to their reliance on well-defined structural dependencies and process logic. On the other hand, supervised learning techniques tend to achieve higher predictive accuracy by capturing more complex patterns and temporal dependencies within the data. Despite their strong performance—especially when large volumes of high-quality historical data are available—such approaches are often less transparent, making the interpretation of the results a more challenging task. [4]

Types of input information

It is common to distinguish several types of input data that are either used to build explicit process models or to generate features for machine learning tasks [35]: *(i) control flow*, which refers to the sequence of events. This captures the order in which actions are carried out within a process—for example, the sequence of steps a patient follows during a hospital visit; *(ii) structured attributes.*: this usually consists of timestamps and additional attributes such as the identity of the person who performed the action or specific parameters related to that event. (timestamps, resources); *(iii) unstructured text*: this might include medical notes or other narrative descriptions that accompany events and provide valuable context for predicting future outcomes.; *(iv) contextual data* about the process itself, such as resource availability, equipment usage, or waiting times. These external conditions can significantly influence how and when process steps unfold, and therefore are crucial for improving prediction accuracy.

In practice, researchers increasingly combine multiple types of input data to train models that can handle this variety of information. Choosing how to represent these inputs remains a central challenge in Predictive Process Monitoring [**?**], as it has a direct impact not only on the accuracy of the model but also on its interpretability.

2.4 Graph Neural Networks in PPM

GNN-based PPM approaches transform each prefix $\mathrm{hd}_k(\sigma)$ into a directed, typed graph whose nodes represent events and whose edges capture control-flow or data-flow relations [9,27,29]. Message-passing layers—such as GCN or GAT—aggregate information from neighbouring nodes, enabling the network to exploit sequential and structural dependencies simultaneously. The rich internal representations produced by GNNs, however, pose new challenges for explanation; we revisit this issue in Section 5, where we introduce the *Type-Aware PGExplainer* (TAPGExplainer).

The formal apparatus developed in this section underpins the prediction models and explanation techniques presented in the remainder of the thesis.

3 Literature Review

The literature on Predictive Process Monitoring (PPM) is extensive. Rather than enumerating every contribution, we trace the chronological evolution of the most influential ideas, grouped by the primary modelling paradigm. Within each strand we highlight milestones that redirected the field; comprehensive surveys provide exhaustive taxonomies elsewhere.

3.1 Next Activity Prediction

Next-activity prediction is the most extensively studied PPM task because it supports resource allocation, SLA compliance, and early-warning dashboards.

Early modelbased era (2008–2013). The earliest mentions of Predictive Process Monitoring (PPM) date back to 2011, when researchers primarily focused on time prediction tasks [7]. At that stage, the dominant approaches were based not on machine learning but on explicit process modeling techniques, such as stochastic Petri nets and discrete-time Markov chains (DTMCs). While transparent, these methods assumed perfectly specified models and degraded on noisy logs.

Supervised feature learning (2014–2016). The seminal work of Maggi *et. al.* [12] reframed PPM as a supervisedlearning problem: ngram and counter features fed decision tree ensembles, eliminating the need for manual modelling. Although the method itself was straightforward, it set a new trend in the field: a shift from manual process modeling to automatic feature construction.

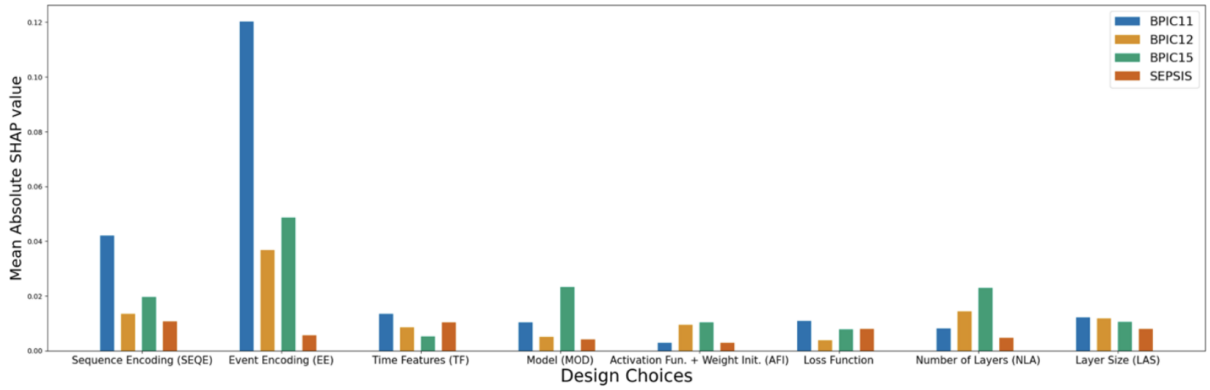
Deeplearning wave Deep-learning approaches gained momentum after the work of Tax et al. *et al.* [34], who trained an LSTM on one-hot activity vectors augmented by three temporal signals (time since last event, time of day, day of week). Across four public logs (BPI-12, Helpdesk, Sepsis, Financial Log), the model improved next-activity accuracy by up to 15 pp over Markov chains and decision trees while simultaneously predicting the next timestamp, complete suffix, and remaining time. Their systematic ablation study showed that temporal features contributed roughly one third of the overall gain, an insight that has guided almost all subsequent sequence models. Motivated by this success, dozens of studies experimented with recurrent, convolutional, and auto-encoder variants. For example, Francescomarino *et al.* [13] introduced an LSTM-based approach that leverages a-priori process knowledge to enhance predictions 9, Evermann *et al.* [11] applied LSTM networks to model temporal dependencies in event logs 5, while Navarin *et al.* [25] explored stacked LSTM architectures for remaining time prediction 17. Researchers also experimented with 1-D convolutions for activity sequence analysis (Aljebrni *et al.* [2]) 6, RNNs with non-sequential control flow handling (Metzger *et al.* [24]) 5, memory-augmented networks to capture long-term dependencies (Mehdiyev *et al.* [23]) 13, and hybrid CNN-LSTM models for spatiotemporal feature extraction [18, 20].

A notable enhancement came from Hinkka *et al.* [16]. They replaced the LSTM with a GRU and introduced *clustered attribute embeddings*: non-categorical event attributes are first grouped via X-means, then represented by fixed-length vectors concatenated to the activity input. On five benchmark logs this design raised next-activity accuracy by up to 6 pp and shortened training time by 35 %.

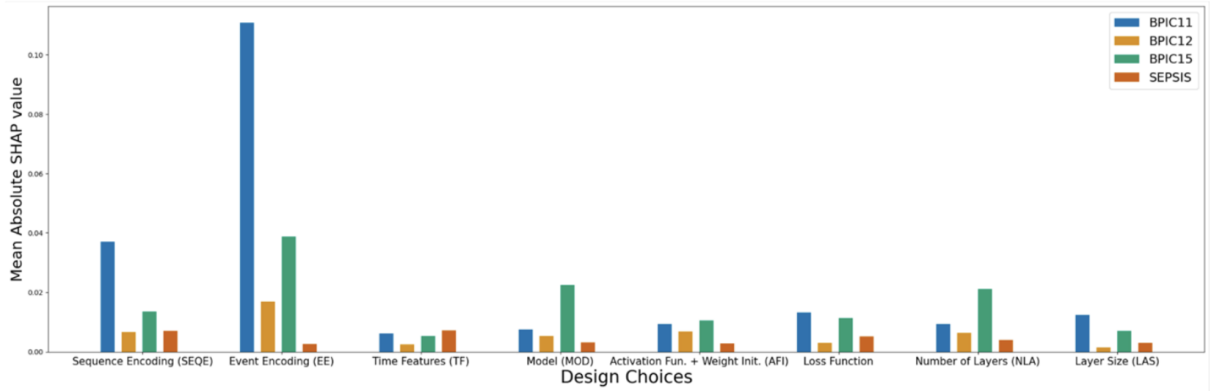
Finally, the large-scale benchmark of Camargo *et al.* [10] provided a rigorous head-to-head comparison of ten deep-learning architectures over twelve public logs using nested cross-validation and Bayesian ranking statistics. RNN-based models—including the original LSTM [34] and the GRU variant [16]—consistently occupied the top tier, clearly outperforming classical machine-learning baselines in both accuracy and MAE, and establishing a de-facto standard evaluation protocol for subsequent PPM research.

3.2 GNNs in PPM

Data representation Data representation plays a critical role in both the accuracy and the interpretability of predictive models. As highlighted in [**?**] (Fig. 2), the way in which both attribute features and event sequences are encoded is one of the key factors in building reliable and precise models. In the context of process mining, data representation and encoding remain particularly challenging due to the complex and structured nature of the data.



(a) Regressor: XGB.



(b) Regressor: EBM.

Figure 2: Mean absolute SHAP values for next activity prediction for different event logs.

One of the essential components to consider is control-flow—that is, the order of activities and the specific path the process follows. Understanding the control flow is crucial for identifying patterns, bottlenecks, deviations, and potential areas for improvement within business processes. At the same time, many analytical tasks also benefit from incorporating data-flow, which focuses on how data is generated, processed, and passed between different activities in the process [4].

In 2022, a study [17] was published that explored various methods of encoding process traces, ranging from traditional one-hot encoding (OHE) to more advanced graph-

based encoders. Although the study itself was not directly related to Predictive Process Monitoring (PPM), it clearly demonstrated the superiority of embeddings derived from process modeling–specific techniques and graph neural networks. These findings highlighted the significant role that data representation plays in understanding the broader context of process behavior, particularly within PPM tasks.

GNN approaches Building on this idea, 2023 introduced PetriNet2Vec [6], a general framework for vectorising process models. In this approach, the authors conceptualize each causal path within a Petri net as a ”sentence” and train a Doc2Vec-like network to generate embeddings for both the entire process model and for individual transitions.

In the past two years, research on graph neural networks has expanded considerably. One of the key works from 2023 is *Embedding Graph Convolutional Networks in Recurrent Neural Networks for Predictive Monitoring* (referred to hereafter as **TACO**). This method constructs a graph-based embedding from an enriched Petri net, which is then combined with attribute-level features and passed into an LSTM model. The **TACO** approach achieved more than a 10 pp performance improvement, underlining that leveraging the inherent structure of a prefix can yield meaningful benefits—albeit with increased computational costs during training.

Around the same time, another study titled *Remaining Cycle Time Prediction with Graph Neural Networks for Predictive Process Monitoring* (referred to here as **Duong**) was introduced [9]. This work proposed the creation of a heterogeneous graph in which each node and edge type is encoded based on its role within the process. The resulting graph is then fed into a graph neural network (GNN) with a recurrent (GRU) component integrated into its architecture. Although the paper does not provide extensive comparisons to firmly establish superiority over other methods, the proposed model still demonstrated solid performance and outperformed many existing approaches.

In 2025, the **PROPHET** [27] framework was introduced, expanding on the ideas from **Duong**. **PROPHET** also generates a heterogeneous graph where each prefix of a process execution is represented as a graph with multiple node and edge types. Nodes capture different event characteristics—such as activity attributes, resource involvement, and timestamps—while edges represent the various relationships between these elements. To enhance predictive accuracy, **PROPHET** employs a Graph Attention Network (GAT) layer over the graph structure. According to the authors, this architecture outperformed most existing models, including **TACO**, across nearly

all tested event logs.

In parallel with these developments, numerous researchers have continued to improve recurrent neural networks, introducing architectures like ATT-BLKAN [39], experimenting with advanced convolutional models such as AMCCN [3], and even applying large language models like LUPIN [26]. However, despite their technical novelty, none of these approaches has provided conclusive evidence to position itself as a breakthrough compared to prior works.

In conclusion, the volume of research on predictive process monitoring continues to grow rapidly. In particular, in the past 2 to 3 years, approaches based on graph neural networks have gained significant traction, consistently outperforming traditional methods by leveraging the structural richness of process representations. These advancements suggest that graph-based modeling is becoming a central direction for the future of predictive process analysis.

3.3 Explainability in PPM

Explainability remains a major challenge in predictive process monitoring.

The importance of explainable monitoring is now widely recognised. The *Process Mining Handbook* devotes an entire section to the topic [35], and recent surveys rank explainability and causality among the field's most pressing themes [5]. Figure 3 reproduces their user study, showing explainability's high perceived value. Similarly, a user evaluation by Di Francescomarino *et al.* found that transparent models foster trust and actionable insights [31].

Challenge	Direction	Average Rating	Standard Deviation
C1. Improve models for humans	D1.1 Interpretability	5.5	0.91
	D1.2 Prescriptive/causal	5.59	1.05
	D1.3 Privacy and fairness	5.0	1.15
C2. Improve model autonomy and standardization	D2.1 Automated creation/tuning	4.77	1.11
	D2.2 Automated adaptation	4.86	1.16
	D2.3 Standard evaluation	5.54	1.33
C3. Improve industry impact	D3.1 Data scarcity	5.95	0.99
	D3.2 Tooling	4.36	1.17
	D3.3 Industrial cases	5.45	1.22

Figure 3: Average Importance Rating and Standard deviations for PPM future directions.

Researchers have incorporated eXplainable AI (XAI) techniques in several ways.

A straightforward way to maintain transparency is one-hot encoding (OHE) [18]; yet its sparsity often harms accuracy. To improve performance, researchers adopt denser schemes such as Count2Vec or learned embeddings, but these reduce visibility into what each dimension represents.

Model-specific efforts tailor explanations to particular architectures—for example, attention-based LSTMs [38], T-gated networks [15], or Bayesian nets—partially address this by highlighting salient attributes during prediction. Model-agnostic, post-hoc methods are equally common: Galanti *et al.* [14] applied SHAP [33] to produce per-instance contribution tables and temporal heatmaps, while Rizzi *et al.* [32] used similar interpretations to refine model accuracy. At the event level, LIME [30] and SHAP [21] have been adapted to quantify feature relevance within each prefix. Graph neural networks offer a promising alternative, because their embeddings integrate control- and data-flow information in a single structure, potentially aiding interpretation. However, despite this fact, effective explainability methods for graph neural networks have not yet been proposed, creating a significant gap and necessitating further development of explainable predictive process monitoring.

Despite this progress, most studies either rely on generic tools best suited to well-structured feature spaces or depend on ad-hoc attention weights that do not guarantee faithful explanations. No comprehensive benchmark yet compares XAI methods across multiple PPM models, and type-aware explanations for GNN-based predictors are still absent.

A rigorous, standardised evaluation of explanation quality is therefore essential—an issue we tackle in Section 5 with the proposed *Type-Aware PGExplainer*.

4 Research Gap & Objectives

4.1 Research Gap

Despite significant advancements in predictive process monitoring (PPM), critical gaps persist in reconciling the accuracy of state-of-the-art models with the need for interpretable and actionable explanations.

Firstly, generic post□hoc techniques such as SHAP and LIME remain ill□suited to PPM: they assume static, tabular features and therefore fail to capture the sequential, temporal and structural dependencies that event logs exhibit [**?**, **?**]. Empirical studies report faithfulness scores below 0.4 and low usability ratings for such methods in real□world logs.

Secondly, the emergence of graph neural networks (GNNs) in PPM—exemplified by frameworks like TACO [29], Duong [9], and PROPHET [27]—has introduced models that leverage graph representations of process traces. Although these architectures achieve superior predictive accuracy by encoding structural and semantic relationships, comparative analyses of explanation methods for GNNs remain absent in the literature, while current GNN explainers lack domain-specific adaptation.
Finally, the field lacks standardized benchmarks and comparative methodology to rigorously evaluate explanation quality in PPM. Current studies assess faithfulness or usability in isolation, making cross-method comparisons impractical [14,32]. This absence of unified metrics and evaluation protocols impedes the development of robust, process-aware XAI techniques. Moreover, while GNNs' structural embeddings hold potential for interpretability, no prior work systematically exploits node/edge semantics to generate type-aware explanations tailored to PPM's unique requirements. Accordingly, the overarching research problem can be stated as follows:

> How can we provide *universal, type aware* XAI explainer for heterogeneous GNN predictors in PPM and *rigorously compare* it to existing methods on a unified benchmark?

4.2 Scope of This Study

Based on the findings of the previous study, this study has **3** main goals:

1. *to conduct a fundamental analysis and comparison of graph neural network (GNN) models from the perspective of accuracy and explainability*. GNNs have already demonstrated high predictive accuracy, and in theory, they also offer potential for interpretability.

2. *to propose a method for explainable predictive process monitoring*. Unfortunately, there is no work, which is practically exploring GNN's explanation methods and therefore there are no PPM oriented solutions. This work introduces the first process oriented explainability method: **Type-Aware PG-Explainer**. This approach reimagines the PGExplainer model, retaining its strengths (discussed in detail later) while addressing its limitations.

3. *to create a benchmark for future researches in the field of GNN Explainability*. Since no prior research has systematically explored this topic within the context of Predictive Process Monitoring (PPM), this work also aims to fill that gap.

To do this, three of the most widely used and powerful GNN-based models were chosen as the basis for evaluating whether existing explainers could be effectively applied to predictive tasks in process monitoring.

5 Proposed Method

5.1 Overall pipeline diagram

This thesis targets *next activity prediction* based on supervised-learning techniques (more specifically: GNN). The next activity prediction is one of the most popular targets among all the researches: more than 70% consider this task as the main one. As for the technique, the choice of a supervised-based approach—specifically, graph neural networks—is driven by two key factors: the strong predictive capabilities 3.2 of these models and their potential advantage 3.3 in terms of explainability. This explainability stems from the way information is structured and how embeddings are generated within graph-based architectures, which often makes them more transparent compared to many other methods.
This section presents a comprehensive analysis of contemporary approaches to predictive process monitoring (PPM), assessing *both* their predictive performance and their explanatory capacity. The investigation is organised into three main phases:

1. **Predictive Model Evaluation**
   we first critically examine state-of-the-art graph neural network (GNN)-based methods in PPM. These models are analyzed for their functional efficacy in real-world prediction tasks, with particular emphasis on their architectural suitability for temporal process data.

2. **Explainability techniques**
   Subsequently, the study shifts focus to the interpretability of GNN-driven predictions. A systematic review of existing GNN explanation techniques is undertaken, prioritizing widely adopted methodologies within the XAI (Explainable AI) literature. Through comparative analysis, we identify inherent limitations in generic GNN explainers when applied to PPM contexts—notably, their inability to account for process-specific semantics (e.g., activity types, temporal dependencies).

   To bridge this gap, we introduce **Type-Aware PGExplainer**, a novel framework extending the PGExplainer architecture. This adaptation retains the orig-

inal model's strengths in graph-structured interpretability while incorporating domain-aware mechanisms to address PPM-specific explanatory challenges.

3. **Experimental Study**
   Later sections detail the experimental protocol for benchmarking explanation methods, including metrics for fidelity, robustness, and human interpretability. Quantitative and qualitative results are presented to substantiate the superiority of our proposed approach in PPM scenarios.

5.2 Predictive Model Evaluation

In this section, we will introduce the GNN-based models that were identified during the review of current popular methods. It is worth noting that there are other approaches using graph neural networks within the broader field of Process Mining. However, many of them are either not focused specifically on the Predictive Process Monitoring (PPM) context, or they lack reproducibility due to the absence of publicly available implementations (e.g., no code on GitHub). The three selected models represent a strong and representative sample, as they are among the most powerful and widely recognized methods in this area.

TACO

The method proposed by Rama-Maneiro [29] combines information from the event log with structural knowledge extracted from a process model, resulting in a hybrid representation of the prefix. The approach begins by applying a process discovery algorithm to generate several candidate Petri nets from the event log, selecting the most suitable one. This selected net is then transformed into a place graph, where the nodes represent places in the Petri net, and the edges capture the connections between them. From this graph, an adjacency matrix and a normalized Laplacian matrix are computed. These, along with a node feature matrix, are fed into a recurrent graph convolutional neural network (GRNN).
The node feature matrix is constructed using token replay: the current case prefix is replayed through the Petri net, and for each place, the number and timing of token visits are recorded. This results in a numerical feature representation for each node. The GRNN processes the pair of Laplacian and node features and uses max-pooling to produce a compact graph embedding.
In parallel, four attributes are extracted from the prefix itself for each event: activity

type, time since the previous event, time since the start of the case, and additional event-level features. These form a prefix feature matrix. The graph embedding is then concatenated with this matrix and passed to an LSTM layer, followed by a fully connected softmax layer that predicts the next activity.
In this way, the model captures both the structural elements of the Petri net and the sequential dynamics of the prefix, leading to improved predictive accuracy.

Prophet

The PROPHET [27] method aims to predict the next activity by integrating multi-view information about the process prefix into a single heterogeneous graph. This graph is constructed from the event log, where nodes of different types represent various process entities—such as the events themselves, their attributes (e.g., cost, urgency), resources involved, and timestamps. Multiple edge types capture relationships like event sequence, event-to-attribute links, shared resources, and others. This structure allows both contextual and structural dependencies to be explicitly encoded. Each node is assigned a feature vector: categorical values are transformed via one-hot encoding or embeddings, numerical features are normalized, and time-related nodes are enriched with sine-cosine components to capture temporal cycles. The resulting graph is then processed using a multi-head Graph Attention Network (GAT). For each edge type, separate attention scores are computed, which guide the aggregation of messages from neighboring nodes—enabling the model to focus on the most relevant connections between heterogeneous elements of the prefix.
The output embeddings of all nodes are passed through a global attention-based read-out layer to generate a compact representation of the entire prefix. Additional aggregated statistics, such as trace length or elapsed time, can be concatenated to this vector if needed. Finally, the combined representation is passed through a fully connected softmax layer that outputs a probability distribution over the possible next activities.

Duong

The method [9] predicts the remaining time of a case by representing the process prefix as a directed event graph. Each node in the graph corresponds to an event and carries a feature vector composed only of the activity and timestamp attributes. The activity is encoded as a one-hot vector, while the timestamp is broken down into four numerical features: time since the previous event, time since the case started, time of day, and day of the week. Edges between events are categorized into three semantic

types: forward (transition to a new activity within the same case), backward (return to a previously seen activity), and repeat (connection between two occurrences of the same activity). Thanks to this structure, the GNN can process prefixes of any length without requiring the artificial padding that LSTM-based approaches typically need. The model applies a standard message-passing scheme on the graph, but replaces the classic UPDATE function with a GRU cell, allowing each node to accumulate the history of received messages. At layer l+1, each node we receives an aggregated message:

$$h_i^{l+1} = \text{GRU}(m_i^{l+1}, h_i^l)$$

where h_j^l is the hidden state of neighbor j, W is a transformation matrix, and e_{ji} is a scalar weight representing the importance of the edge from j to i. The new hidden state of node i is then computed using a GRU:

$$m_i^{l+1} = \sum_{j \in \mathcal{N}(i)} e_{ji} \cdot W h_j^l$$

After a fixed number of layers, the model performs global mean pooling over all node embeddings to obtain a single vector representation of the entire prefix graph. This compact embedding is passed through a multilayer perceptron, which outputs the estimated remaining time.

The choice of the model

The three models were initially selected as state-of-the-art representatives within their respective domains. However, to rigorously evaluate the explanatory capabilities of interpretability methods, a single model required prioritization. All three approaches were compared with each other and one of the most popular RNN solution [34]. The same event logs were used across all experiments, and each dataset was split into 80% for training and 20% for testing. To ensure an objective evaluation, the primary metric used was average accuracy, computed across all trace lengths. This setup made it possible to directly compare the performance of the models under identical conditions, highlighting their strengths and weaknesses in a controlled environment.

Following comprehensive benchmarking across datasets (see Section 6) and architectural analysis, Prophet was ultimately chosen as the optimal candidate for this study. Prophet, a recently proposed framework (see Table 2), demonstrates superior performance in predictive accuracy while maintaining computational efficiency. Comparative analysis revealed two primary candidates: TACO and Prophet. While Duong

Dataset	TACO	PROPHET	Duong	Tax
Helpdesk	**0.783**	0.757	0.641	0.723
BPI-2012-A	0.776	**0.781**	0.601	0.612
BPI-2012-O	**0.809**	0.759	0.709	0.799
BPI-2012-W	0.647	**0.791**	0.650	0.789
EnvPermit	0.853	**0.871**	0.840	0.903

Table 2: Comparison of model performance across datasets

et al.'s approach provided foundational insights, its predictive robustness proved inadequate for multi-attribute log datasets.

Between TACO and Prophet, the former emphasizes structural process features through a hybrid architecture: graph neural networks (GNNs) generate embeddings, which are concatenated with attribute-based features and processed by an LSTM layer. This multi-stage design introduces significant interpretability challenges. Preliminary attempts to apply a two-stage SHAP-based interpretability framework yielded negligible insights due to the model's inherent complexity.

In contrast, Prophet achieves comparable accuracy—particularly in multi-attribute scenarios—while offering inherent interpretability. This tradeoff aligns with the study's emphasis on actionable explanations, rendering Prophet the pragmatically superior choice for explainable predictive process monitoring.

5.3 Explainability techniques

The primary objective of this study was to systematically evaluate and compare interpretability methods within the XAI (Explainable Artificial Intelligence) domain to identify optimal approaches for predictive process monitoring (PPM).

Existing XAI methodologies are broadly categorized into distinct families: gradient-based techniques, model-agnostic perturbation methods, and surrogate models (globally or locally scoped). While alternative frameworks exist, they were excluded from this analysis due to either limited applicability to PPM contexts or architectural incompatibility with our proposed solution.

To refine the scope, this investigation prioritizes local explanation methods, as their ability to elucidate individual predictions aligns with the decision-critical requirements of PPM. Such granular interpretability is essential for fostering trust and enabling actionable insights in operational settings [31].

From a methodological standpoint, surrogate methods exhibit inferior performance in generating local explanations compared to model-agnostic perturbation-based ap-

proaches. While gradient-based explainers are computationally efficient, their reliance on gradients derived from shifted embedding-space coordinates—rather than meaningful node and edge interactions—limits their utility. These methods primarily capture model sensitivity to micro-perturbations in feature space, failing to isolate causal subgraphs critical for domain-aware interpretability in predictive process monitoring (PPM).

To address this, our comparative framework incorporates three perturbation-driven methods (GNNExplainer, PGExplainer, SubgraphX) and one architecture-specific technique (Attention Top-k), the latter tailored to Prophet's design.

The subsequent analysis shows each method's strengths and limitations within PPM contexts. Building on these insights, we propose **Type-Aware PGExplainer**, an extention of the current PGExplainer method. This adaptation bridges the gap between generic GNN explanation frameworks and the unique spatiotemporal demands of process mining.

GNNExplainer

This is a classic post hoc method that optimizes a continuous mask to identify the minimal subgraph—nodes, edges, and/or features—necessary to preserve the original prediction made by the GNN. GNNExplainer was introduced by Rex Ying *et al.* [41] as the first architecture-agnostic, post-hoc explanation technique for GNNs; the authors validated it on the synthetic datasets BA-Shapes, BA-Community, and Tree-Cycles, as well as on the real-world MUTAG and Reddit-Binary graphs, visualising both node- and graph-level model decisions.

The method learns a continuous mask by solving

$$\max_{G_s} \mathrm{MI}\left(Y, (G_s, X_s)\right) = H(Y) - H\left(Y \mid G = G_s,\ X = X_s\right) \tag{1}$$

thereby minimising the conditional entropy; the final sub-graph is constrained by a size regulariser and highlights the components that contribute most to the prediction. The method can be adapted to heterogeneous structures by training separate masks for each type of edge or node, or by introducing weighting schemes per type. This adoption is enough to run through Prophet model.

PGExplainer

PGExplainer—short for *Parameterized Graph Explainer*—was introduced by Luo *et al.* at ICLR 2021 as a method for producing *collective* explanations across many input graphs, yielding up to a 24% AUC improvement over GNNExplainer on MUTAG and other benchmarks [22].

Instead of selecting important edges separately for each instance by optimization (as in GNNExplainer, for example), PGExplainer trains a separate explanatory network (Multi-layer perceptrone) to predict the probability that each edge belongs to an important subgraph of the explanation. MLP predicts the edge-importance probabilities $p_\theta(e_{ij})$, thereby defining an approximate distribution $q_\theta(G_S)$ over candidate explanatory sub-graphs.

Training proceeds by the re-parameterisation trick and minimises the mutual information between the explanation and the model's prediction, which is equivalent to GNNExplainer optimization task.

The result is a concise set of edges that, collectively, contributes most to the model's decision while remaining easy to visualise and interpret.

Attention Top-k

This technique is conceptually straightforward and incurs minimal computational overhead. If the base PPM model uses attention-based architectures such as GAT, Transformer, or Performer, the attention scores can be interpreted as rough indicators of feature or node importance. By selecting the top k nodes or edges with the highest attention weights, a lightweight explanation can be produced with minimal computation. This method only works when attention is already part of the model; for standard GCN or LSTM architectures, the model must be modified. Therefore, this method is not applicible to TACO model. On heterogeneous graphs, it provides attention weights per edge type, which is helpful for analysis. However, it's important to note that high attention weights do not necessarily imply causal importance, so further validation using fidelity or ablation methods is recommended.

SubgraphX

SubgraphX, proposed by Yuan *et al.* at ICML 2021, is the first explainer that explicitly searches for a *connected* sub-graph whose contribution to a GNN prediction is maximised [42].

The method frames explanation as a Shapley-value problem and applies Monte-Carlo Tree Search (MCTS) to expand or prune nodes according to their marginal gain, converging on a compact, high-impact motif.
For a target sub-graph G_i within the player set P, its contribution is scored by

$$\phi(G_i) = \sum_{S \subseteq P \setminus \{G_i\}} \frac{|S|! \cdot (|P| - |S| - 1)!}{|P|!} \left(f(S \cup \{G_i\}) - f(S) \right) \tag{2}$$

where $f(\cdot)$ is the model's prediction.
MCTS balances exploration and exploitation, evaluating sub-graphs in an L-hop neighbourhood; this drastically accelerates the search while preserving Shapley's theoretical guarantees.
Conceptually, the algorithm seeks

$$\max_S \phi(S) = \sum_{T \subseteq S} \frac{(-1)^{|T|-|S|}}{|T|} \left(f(T) - f(T \setminus \{v\}) \right) \tag{3}$$

with S the explanatory sub-graph and v a candidate node.

5.4 Type-Aware PGExplainer

Following a systematic evaluation of contemporary methods and their operational viability, the **Type-Aware PGExplainer** emerged as the optimal solution for explainable predictive process monitoring. This framework extends the foundational PGExplainer architecture, enhancing its adaptability to process-centric data through domain-specific modifications.
The core innovation of PGExplainer lies in its ability to train a universal mask generator — unlike approaches such as GNNExplainer, which require subgraph mask optimization for individual graph instances. This capability eliminates the need for instance-specific training, a critical advantage for scaling interpretability to real-world business process data and enabling autonomous deployment in operational settings.
However, despite these computational efficiencies, the original PGExplainer exhibits architectural limitations when applied to heterogeneous graph structures. Its design lacks mechanisms to account for different type semantics, which is used in Prophet heterogeneous graph and inherent in process mining datasets. These shortcomings restrict its ability to isolate causally relevant subgraphs in multi-attribute process logs, where node and edge heterogeneity exists.

The basic idea behind TAPG is to account for different types of edges by paying separate attention to each type of edge. While PGExplainer treats all edges as a single rib type, TAPGExplainer distinguishes between rib types and takes them into account in the explanation. To accomplish this, several key changes are introduced:

1. **Forming edge features:** for each edge $e = \left(u \xrightarrow{t} v\right)$ of type $t \in \mathcal{T}$ we take the concatenation of the end embeddings: $x_e = [h_u \,\|\, h_v]$, where h_u, h_v embeddings of edges u, v respectively.

2. The learning objective: As in PGExplainer, TAPG relies on the same information-theoretic backbone: the input graph G_o is decomposed into an explanatory subgraph G_s and a noise part ΔG. The goal is to pick G_s that maximises mutual information with the model prediction; the objective stays

$$\max_{G_s} \mathrm{MI}(Y_o, G_s) = H(Y_o) - H(Y_o \mid G = G_s). \tag{4}$$

   Because $H(Y_o)$ is constant, we minimise the conditional entropy

$$\min_{G_s} H(Y_o \mid G = G_s). \tag{5}$$

   Exhaustive search is infeasible, exactly as before we introduce a stochastic mask: each edge (i, j) is a binary variable e_{ij} with inclusion probability θ_{ij}. The full graph distribution becomes

$$P(G) = \prod_{(i,j)\in\mathcal{E}} P(e_{ij}), \quad P(e_{ij} = 1) = \theta_{ij}. \tag{6}$$

   The optimisation is rewritten as an expectation over a Gilbert (Bernoulli) random graph:

$$\min_{G_s} H(Y_o \mid G = G_s) \approx \min_{\Theta} \mathbb{E}_{G_s \sim q(\Theta)} \left[H(Y_o \mid G = G_s) \right], \tag{7}$$

   where $q(\Theta)$ is the subgraph distribution induced by the edge-inclusion probabilities θ.

3. The reparameterization trick: to handle the discrete mask, we keep the same Binary Concrete relaxation used in PGExplainer. Each edge indicator e_{ij} is replaced by a continuous sample $\hat{e}_{ij} \in (0, 1)$:

$$\epsilon \sim \text{Uniform}(0,1), \qquad \hat{e}_{ij} = \sigma\big((\log\epsilon - \log(1-\epsilon) + \omega_{ij})/\tau\big), \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid, ω_{ij} is a learnable logit and τ controls the softness. As $\tau \to 0$, $\hat{e}_{ij}$ becomes Bernoulli with $P(e_{ij}=1) = \theta_{ij} = \frac{\exp\omega_{ij}}{1+\exp\omega_{ij}}$, so $\hat{G}_s \to G_s$.

The relaxed objective therefore stays

$$\min_{\Omega} \ \mathbb{E}_{\epsilon\sim U(0,1)} H\big(Y_o \mid G = \hat{G}_s\big). \tag{5}$$

Following, we replace the conditional entropy by a cross-entropy that can be estimated with K Monte-Carlo samples:

$$\min_{\Omega} \ \frac{1}{K}\sum_{k=1}^{K}\sum_{c=1}^{C} P_\Phi\big(Y=c \mid G=G_o\big) \ \log P_\Phi\big(Y=c \mid G=\hat{G}_s^{(k)}\big). \tag{6}$$

Note. This entire reparameterisationis exactly the same as in PGExplainer; TAPG changes only how the logits ω_{ij} are produced (edge-type aware) and how loss is calculated.

4. **Generation of type-aware logits:** PGExplainer turns local edge masks into a *global* explainer by sharing one parameterised network g_Ψ across a set of graphs $\mathcal{I}$. we retain this idea, but inject edge–type information into the logit generation step.

   For any input graph G_o with node features X, the backbone GNN first produces node embeddings and a prediction:

$$Z = \text{GNNE}_{\Phi_0}(G_o, X), \qquad Y = \text{GNNC}_{\Phi_1}(Z). \tag{7}$$

   . The explanatory multilayer percentrone (MLP) computes for each edge a logit, a scalar measure of importance of each edge. While in PGExplainer there is one MLP for the whole graph, in TAPG MLP computations occur in parallel for each type of edge $t \in \mathcal{T}$ separately. That is, for each unique link type t, its own MLP is allocated and trained. For an edge e of type t with feature vector $x_e = [\, z_u;\, z_v \,]$ I compute

$$\omega_e^{(t)} = f_t(x_e), \qquad f_t : \mathbb{R}^{2F} \to \mathbb{R}, \tag{8}$$

and collect the logits in $\Omega = \{\omega_e^{(t)}\}_{e\in\mathcal{E}}$. When $|\mathcal{T}| = 1$ this reduces to the original PGExplainer. This approach allows to adjust for each type of link separately and account for different types of links differently. This is especially useful for process log semantics, since there the link types determine the structure of the process.

Following PGExplainer, the explanation network is trained *collectively* on $\mathcal{I}$ via a Monte-Carlo estimate of the cross-entropy between the original prediction Y and the prediction obtained on a sampled explanatory graph $\hat{G}_s^{(i,k)}$:

$$\min_{\Psi} \; -\sum_{i\in\mathcal{I}} \frac{1}{K} \sum_{k=1}^{K} \sum_{c=1}^{C} P_\Phi\big(Y = c \,|\, G = G_o^{(i)}\big) \, \log P_\Phi\big(Y = c \,|\, G = \hat{G}_s^{(i,k)}\big) + \sum_{t\in\mathcal{T}} \lambda_t \|m_t\|_1, \tag{9}$$

where the last term is *new*: a per-type ℓ_1-penalty that keeps masks of frequent edge types from dominating others.

5. **Prediction and losses:** A GNN model with weights (mask) z taken into account (which are computed from logits using concrete-sampling – creating the mask from logits remains the same as in PGExplainer) produces a prediction $\hat{y}$. Let's denote y_{orig} is the original class (which the model predicted without the mask) – it is its behavior that the model wants to preserve. The loss function of TAPGExplainer is similar to PGExplainer, except for the modification in the size-regularization part:

   - **Loss prediction** L_{pred}: similarly calculated as the negative logarithm of the probability of the original class under the new distribution:

   $$L_{\text{pred}} = -\log \hat{P}(y_{\text{orig}} \mid G, z). \tag{8}$$

   The implementation takes the corresponding probability through softmax and averages the minus-logarithm. Minimizing L_{pred} induces the mask to preserve model-critical dependencies.

   - **Per-type** L_{size}: instead of a single global penalty, summarization is done by type:

   $$L_{\text{size}} = \sum_{t\in\mathcal{T}} \lambda_t \sum_{e\,:\,\text{type}(e)=t} |z_e|. \tag{9}$$

Here λ_t is the sparsity factor for type t. The default is $\lambda_t = \lambda$ for all types of t. Then the formula is equivalent to $\lambda \sum_e |z_e|$, but separate summation ensures *even pressure* penalty on different groups of edges. In particular, if one type includes many weak ties and another type includes a few strong ties, the per-type scheme will prevent the former from "spreading" the mask over many edges, avoiding a large penalty. Each type pays a penalty proportional to the total weight of *inside its class*, preventing frequent types from dominating the mask.

- **Entropic** L_{ent}: calculated in the same way as in PGExplainer (binary mask entropy), with the same coefficient μ. This term makes z_e tend to 0 or 1, making the choice of edges clearer. In the context of TAPGExplainer, it can also help ensure that a limited distinct subset of connections is selected for each type, instead of a fuzzy distribution of weights on many edges.

Algorithm The algorithm for TAPGExplainer is shown in Algorithm:

Algorithm 1 TAPG-Train

Input: The input graph $G_o = (\mathcal{V}, \mathcal{E})$, node features $\mathbf{X}$, node labels Y, K - the number of samples.

for each graph $G_o^{(i)}$ **do**
  $Z^{(i)} \leftarrow \text{GNN\_E}_\Phi(G_o^{(i)}, X^{(i)})$
  $Y_o^{(i)} \leftarrow \text{GNN\_C}_\Phi(Z^{(i)})$
end for
for epoch $= 1 \dots T$ **do**
  for each graph $G_o^{(i)}$ **do**
    $\Omega \leftarrow$ edge_logits_TAPG$(Z^{(i)}, G_o^{(i)})$ ▷ Calculates as shown in Eq. **??**
    for $k = 1 \dots K$ **do**
      $\hat{Z}_s^{(i,k)} \leftarrow \text{ConcreteSample}(\Omega)$
      $Y_s^{(i,k)} \leftarrow \text{GNNC}_\Phi(\text{GNNE}_\Phi(\hat{Z}_s^{(i,k)}, X^{(i)}))$
    end for
  end for
  Compute L_{pred} from $\{Y_o^{(i)}\}, \{Y_s^{(i,k)}\}$
  Compute L_{size} per-type
  Compute L_{entropy}
  $\Psi \leftarrow \Psi - \eta \nabla_\Psi \left(L_{\text{pred}} + L_{\text{size}} + L_{\text{entropy}}\right)$
end for

6 Datasets & Metrics

6.1 Datasets

BPI–2012 A (Application view)

A dataset from the BPI Challenge 2012, representing the credit application process from the perspective of sequential activities: application registration, risk assessment, proposal preparation, and loan approval. A key feature is the presence of precise timestamps and frequent looped steps, which are especially valuable for predicting time-related outcomes.

BPI–2012 O (Organizational view)

This is the same credit process as in BPI□2012□A, but each event is annotated with the department or individual employee responsible for the action. This enables the analysis of team workloads and the organizational structure's impact on process per-

formance.

BPI–2013 Closed Problems

These records come from problem management system and include only closed (resolved) problem cases. Each case documents the full chain of diagnostics, corrective actions taken, and final classification of the root cause.

BPI–2013 Incidents

This dataset contains information about incidents (issue reports) registered in the same IT service department. It logs the creation, categorization, escalation, and resolution of each incident, making it useful for studying the factors that influence resolution time.

Helpdesk

This log comes from a real customer support service, recording the opening, escalation, and closure of client tickets. It contains several thousand cases and includes details such as the request type, responsible operator, and timestamps for each operation.

EnvPermit

This dataset describes the process of handling environmental permit applications—from initial submission to the final decision made by the regulatory authority. The records reflect steps such as content checks, expert consultations, and either approval or rejection of the request.

	BPI-2012-A	BPI-2012-O	BPI-2013-CP	BPI-2013-I	Helpdesk	EnvPermit
n. cases	13087	5015	1487	7554	4552	1434
n. activities	10	7	7	13	10	27
n. events	60849	31244	6660	65533	21197	8577
avg. case len	4.65	6.23	4.48	8.68	4.66	5.98
max case len	8	30	35	123	15	25
avg. case duration	8.08	17.19	178.89	12.08	40.85	5.41
max case duration	91.5	89.59	2254.85	771.35	59.99	275.8
n. variants	17	168	327	2278	207	116

Table 3: Summary statistics of event logs

6.2 Metrics

After applying several explainability methods, a key question arises: how can we compare them and assess the quality of the explanations they produce? To address this, the field of Explainable AI (XAI) offers a variety of evaluation metrics—both quantitative (automated) and qualitative (human-involved or based on reference explanations).

Below are the main metrics commonly used in Predictive Process Monitoring (PPM), along with a brief overview of their purpose and whether they apply at the local (individual explanation) or global (dataset-level) scale:

Sufficiency. The fidelity sufficiency F_{suf} is the difference in the predicted probability when computed on the graph and on the explanation.
The metric is defined y:

$$F_{\text{suf}} = \frac{1}{N_t - 1} \sum_{k=1}^{N_t - 1} \left(g(G) - g(G_{\text{exp}}(t_k)) \right), \tag{10}$$

i.e., the average change in prediction over all the possible hard masks.

Comprehensiveness. The fidelity comprehensiveness F_{com} is instead the difference in the predicted probability when computed on the graph and on the complement of the explanation. Proceeding as in the computation of the sufficiency is defined:

$$F_{\text{com}} = \frac{1}{N_t - 1} \sum_{k=1}^{N_t - 1} \left(g(G) - g(G \setminus G_{\text{exp}}(t_k)) \right), \tag{11}$$

where now $G \setminus G_{\text{exp}}(t_k)$ is the complement of the hard mask $G_{\text{exp}}(t_k)$. This metric may as well assume negative values, but good explanations have in this case F_{com} higher scores.

Fidelity Fidelity measures how well an explanation reflects the true behavior of the model. The idea is simple: if the model keeps making the same prediction using only the explained part of the input, the explanation is considered sufficient; if removing that part changes the prediction, it's considered necessary.
To adress this metric to our task and to aggregate F_{com} and F_{suf} into a unique fidelity metric, for graph classification tasks we compute *f1-fidelity* (F_{f1}), which is defined by

$$F_{f1} = 2 \cdot \frac{(1 - F_{\text{suf}}) \cdot F_{\text{com}}}{(1 - F_{\text{suf}}) + F_{\text{com}}}. \tag{12}$$

This is indeed the $f1$ score between F_{com} and $(1 - F_{\text{suf}})$. For graph classification task this metric is used as one of the main instead of F_{com} and F_{suf}.

Sparsity

Sparsity captures how concise an explanation is. Ideally, an explanation should include only the most relevant features, leaving out unimportant ones. For example, if only 5 out of 100 nodes are highlighted, sparsity is 0.05. Low sparsity makes explanations easier to understand but may lower fidelity if important information is removed. Thus, it's often used alongside fidelity to analyze the trade-off between simplicity and accuracy. Like fidelity, sparsity is a local metric that can be averaged across examples and is easy to reproduce.

Plausibility Plausibility evaluates how understandable and reasonable the explanation appears to a human expert. When ground truth is available (e.g., in synthetic datasets), it can be measured directly by comparing the explanation to known relevant features. In real-world cases, plausibility is usually assessed through expert feedback: does the explanation align with domain knowledge or known business rules? While subjective and harder to reproduce, plausibility is essential to ensure that explanations not only make sense to the model but also to humans. But during this resarch the metric is ommited.

Stability. An explanation is considered stable if it remains consistent even when the input graph is slightly altered — for instance, by introducing very small changes to node features or the graph structure [1]. To quantify this, we evaluate the **stability** of graph explanations by observing how the model's predictions respond to such perturbations. Beyond requiring that the model outputs for the original subgraph S_u and its perturbed version $S_{u'}$ are close, we also assess whether the internal model behavior differs only slightly, bounded by a small constant δ. Specifically, we enforce the condition:

$$\|\mathcal{L}_{S_u} - \mathcal{L}_{S_{u'}}\|_p \leq \delta.$$

Here, $\mathcal{L}(\cdot)$ refers to any form of model knowledge like output logits $\hat{y}_u$ or embeddings $\mathbf{z}_u$.
we compute graph explanation **instability** as:

$$\text{GES}\left(\mathbf{M}^p_{S_u}, \mathbf{M}^p_{S_{u'}}\right) = \max_{\forall S_{u'} \in \beta(S_u)} D\left(\mathbf{M}^p_{S_u}, \mathbf{M}^p_{S_{u'}}\right), \tag{13}$$

where D represents the cosine distance metric, $\mathbf{M}^p_{S_u}$ and $\mathbf{M}^p_{S_{u'}}$ are the predicted explanation masks for S_u and $S_{u'}$, and β represents a δ-radius ball around S_u for which the model behavior is same.
In the experiments we will evaluate all the methods with Fidelity (comprehensiveness + sufficiency).

6.3 Benchmark Design: Datasets and Evaluation Metrics

A clear benchmark is still missing in predictive process monitoring (PPM). In this section I give one. I bring together 5 public event logs and four automatic metrics so that future papers can compare results on the same ground.
The six logs selected—BPI-2012-A, BPI-2012-O, BPI-2013-Closed Problems, BPI-2013-Incidents, Helpdesk, and EnvPermit are already the backbone of the PPM literature. LSTM pioneers such as Tax et al. [34] evaluate their dual-headed next-activity/time model on BPI-2012 and Helpdesk, reporting up to 15 pp accuracy gains over Markov baselines. Mehdiyev et al. replicate the improvement on Helpdesk, BPI2012 and BPI2013using using multi-stage DL approach [23], while the large-scale survey by Rama-Maneiro et al. [10]ranks the same pair among the twelve most frequently cited public logs, including: BPI 2012, BPI 2013, HelpDesk, EnvPermit, Nasa and Sepsis. The heterogeneous BPI-2012-O variant is indispensable for resource-aware studies (e.g., Rama-Maneiro's TACO [29]), and both 2013 collections serve as standard stress tests for class imbalance in outcome prediction (Galanti et al. [14], Rizzi et al. [32]). Together the six logs span production, finance, public administration and IT support, cover 7 – 27 distinct activities, and range from short transactional traces to multi-month permit cases (see Table 3). For the ensurance, the table 4 is constructed. According to this table it can be said, that any method passing the suite is likely to generalise across control-flow complexity, organisational perspectives and temporal granularity.

Dataset	Representative publications in which it is employed	# papers
HelpDesk	[8, 10, 26, 27, 29]	5
BPI 2012	[3, 8, 10, 19, 27, 29]	6
BPI 2013	[8, 10, 19, 27, 29]	5
EnvPermit	[8, 29]	2

Table 4: Popularity of publicly available event logs in predictive process monitoring research

Model-agnostic quality indicators. Choosing evaluation criteria is harder than choosing data: the domain lacks large pools of process mining experts able to label ground truth explanations, and business users require results that are both faithful and concise. For that reason the benchmark adopts four fully automated metrics that have proved their worth in graph-explanation studies and, crucially, do not require manual annotation.

Comprehensiveness and Sufficiency quantify, respectively, the loss in confidence when the explanation is removed and the confidence retained when only the explanation is kept; they originate in the GNNExplainer paper and remain the fidelity tests [41].

Their harmonic mean, F_f1, was proposed by Agarwal et al. as a single-number fidelity score for graph explanations [1]; it is adopted here to ease ranking across datasets.

Sparsity is the proportion of masked elements captures parsimony and directly penalises oversized rationales (used by Yuan et al. in SubgraphX [42]).

These metrics are task agnostic, scale to thousands of traces, and—unlike usability polls can be recomputed whenever new models appear. They therefore provide a robust "first-line" benchmark in a field where controlled user studies remain rare and expensive.

By clustering the discipline's most cited public logs and the four most widely accepted automatic metrics into one reproducible package, this section delivers the missing reference point. Future work can extend the suite with additional domains or human centred criteria, yet the baseline offered here already covers the majority of scenarios reported in the last decade and establishes a common yardstick against which new algorithms can be measured.

7 Experimental Study

Github: `https://github.com/lubludrova/EPPM`

Explainability Results

To begin, a comparative review of all GNN models (TACO, PROPHET, DUONG) was done and the model that works best – Prophet – was selected. The steps of the comparison as well as the motivation for choosing the model were explained in the Section 5.2.

Regarding the explainability methods, below is a Table comparing the TAPG method with other different interpretation methods. The *f1-fidelity* (F_{f1}) metric was chosen as the main metric, which is the best fit for comparison. Experiments were run on randomly selected 5 cases from the test sample for each dataset.

Table 5 shows that **TAPGExplainer** occupies the top positions on nearly all process-mining datasets. It ties with Att-Top-k on *BPI-2012-A* (0.34) and attains the best score on the *BPI-2012-O* (0.61), exceeding the next-best method by five percentage points. On the corporate log datasets *HelpDesk* and *EnvPermit*, it matches the baseline GNNExplainer, while PGExplainer and SubX lag behind by as much as 0.20–0.24.

Model \ Explainer	BPI-2012-A	BPI-2012-O	BPI-2013-CP	HelpDesk	EnvPermit
GNNExpl.	0.31	0.48	0.30	**0.43**	0.32
PGExpl.	0.11	0.19	0.28	0.39	0.21
Att-Top-k	**0.34**	0.56	0.31	0.38	0.37
SubX.	0.22	0.11	**0.52**	0.38	**0.38**
TAPGExpl.	**0.34**	**0.61**	0.29	0.37	0.19

Table 5: f1-fidelity Comparison of explainers across different datasets

Table 6 reveals the source of this advantage. TAPG consistently produces the smallest Sufficiency values - 0.06 on *BPI-2012-A* and 0.13–0.15 on *HelpDesk* and *EnvPermit*. Coupled with competitive Comprehensiveness, this drives the harmonic mean upwards, whereas competing methods typically improve only one of the two components and therefore achieve lower overall scores. All results were obtained without dataset-specific hyper-parameter tuning, underscoring the method's robustness.

Model \ Explainer	BPI-2012-A	BPI-2012-O	BPI-2013-CP	HelpDesk	EnvPermit
GNNExpl.	0.12	0.40	0.26	0.38	0.20
PGExpl.	0.4	0.72	0.29	0.41	0.34
Att-Top-k	0.12	**0.25**	0.26	0.46	0.19
SubX.	0.21	0.38	0.44	0.25	0.43
TAPGExpl.	**0.06**	0.32	**0.17**	**0.13**	**0.15**

Table 6: Sufficiency: Comparison of explainers across different datasets

For practitioners, these numbers indicate that the subgraphs extracted by TAPG are both critical to the model's decision and compact enough to almost reproduce the prediction on their own. This rare combination, directly captured by f_1-fidelity, sets TAPG apart from existing explainability techniques.

Experimental results (Picture 4)indicate that SubGraphX incurs prohibitive computational overhead on real event logs. The time of training and practicing the model on one example is dozens of times longer than any other model, moreover, this model has no inference mode. These two facts make the SubgraphX model impossible to use on real business logs.

The experiments were conducted on the hardware: a 14-inch MacBook Pro equipped with an Apple M3 Pro chip, 18 GB of RAM, and macOS Sonoma 14.5.

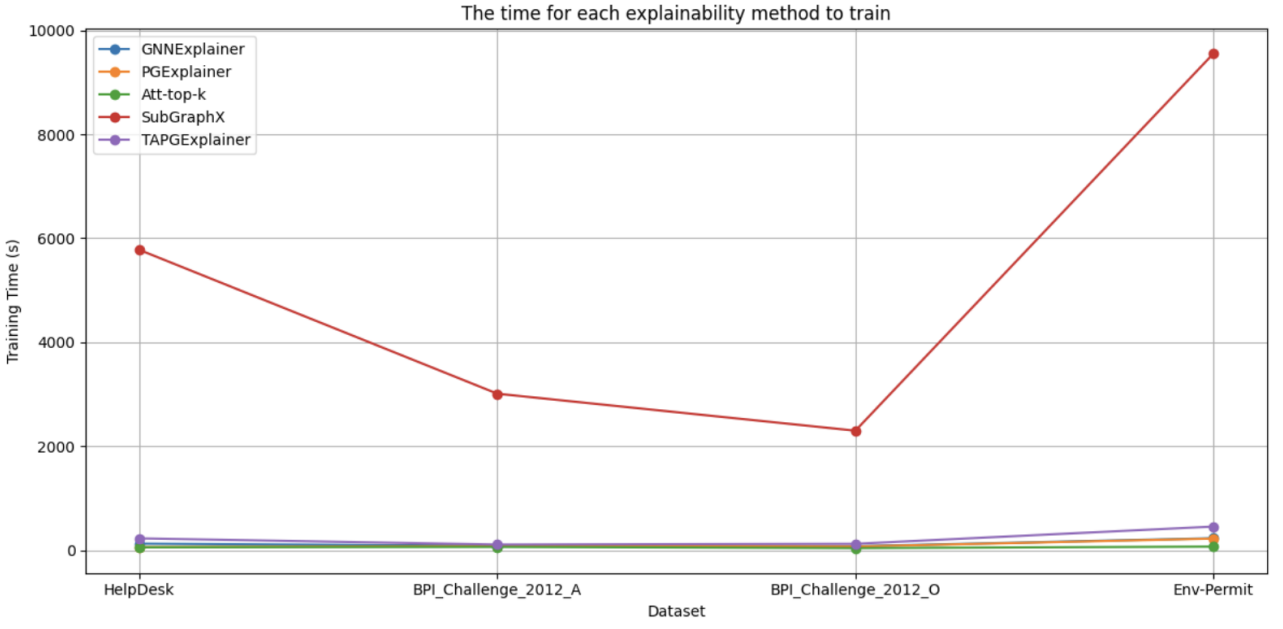


Figure 4: Training time for each explainability model

Figure 4 shows all explainability methods, while Figure 5 shows all methods except SubGraphX for convinience in comparision for other methods. It can be seen that the TAPGExplainer model takes longer to train than the others, but this is not surprising

– this model needs to be trained by several MLP models. But, the advantage of Type-Aware PGExplainer is that this model can run in inference mode, where it can do one-shot explanations – meaning that explanations can be generated without re-training the explainer for each individual input. This significantly reduces the overall time required during deployment and makes these methods more viable for real-world use. Moreover, parallel cores per MLP head can be used for training and in this case training can be faster than even PGExplainer due to parallel computation. This is another advantage for running in a real business infrastructure.

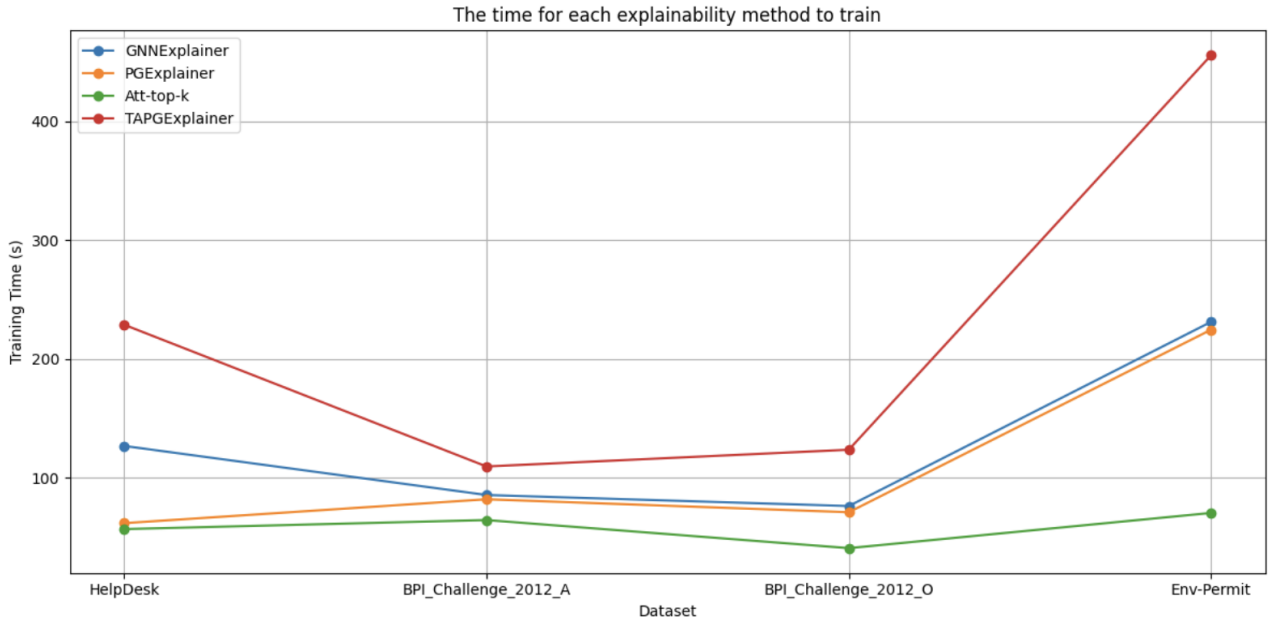


Figure 5: Training time for each explainability model without SubgraphX

Down here there 3 examples of explanations, done by TAPGEXplainer. As you can see, graph structure is really convinient for analyzing business processes.

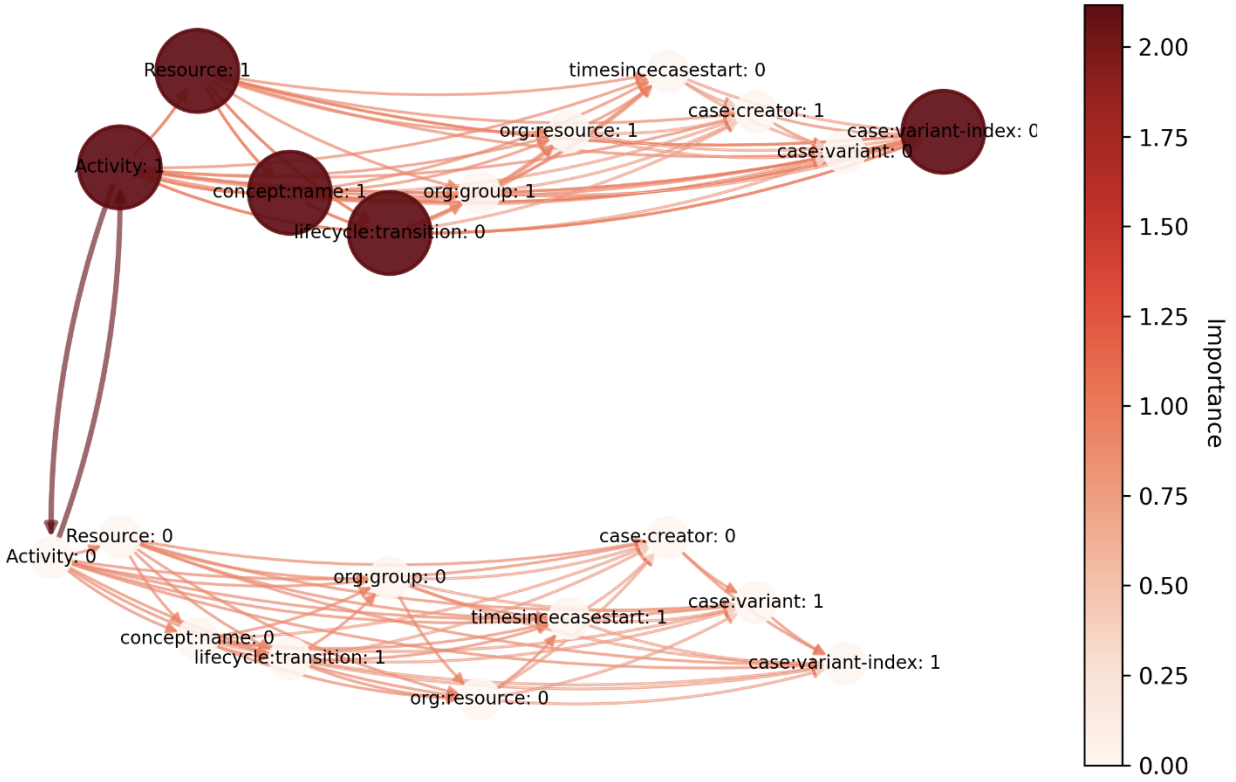


Figure 6: Explanation graph done by TAPGExplainer on BPI2013.

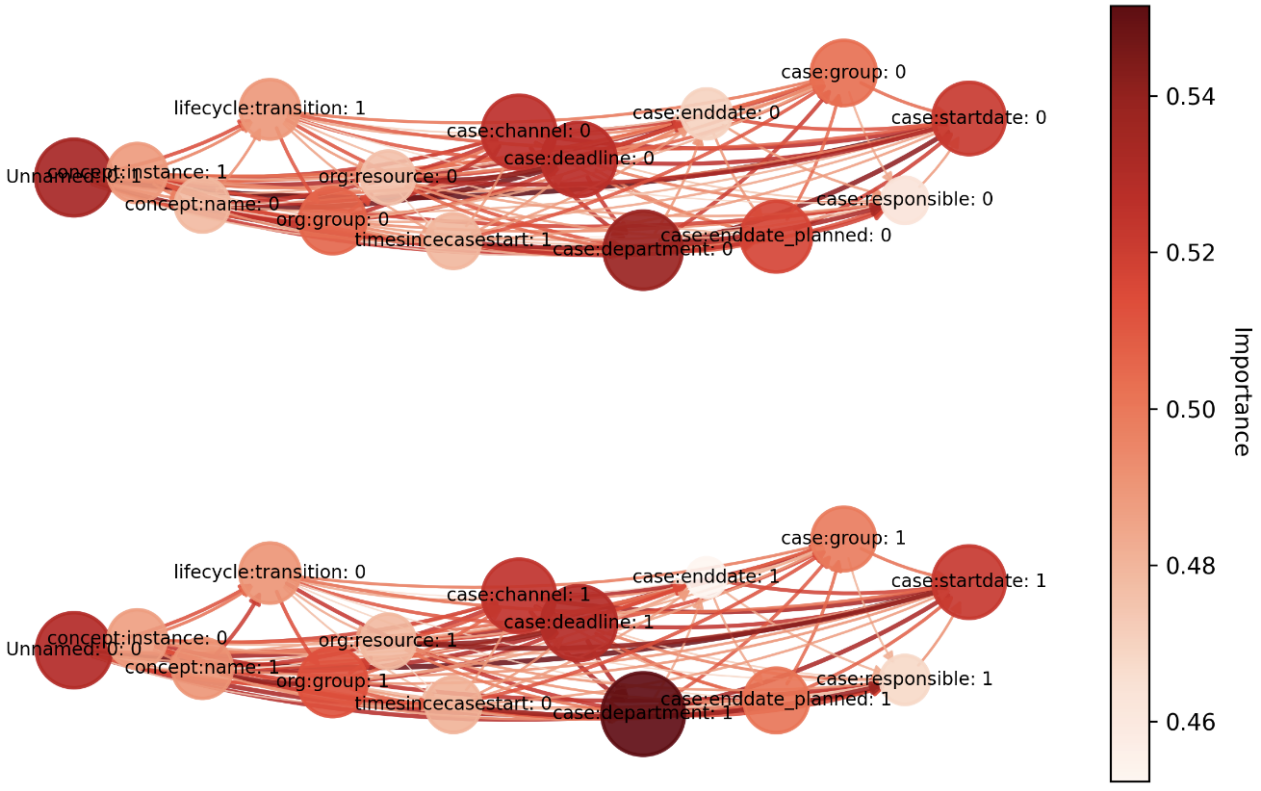


Figure 7: Explanation graph done by TAPGExplainer on BPI2013.

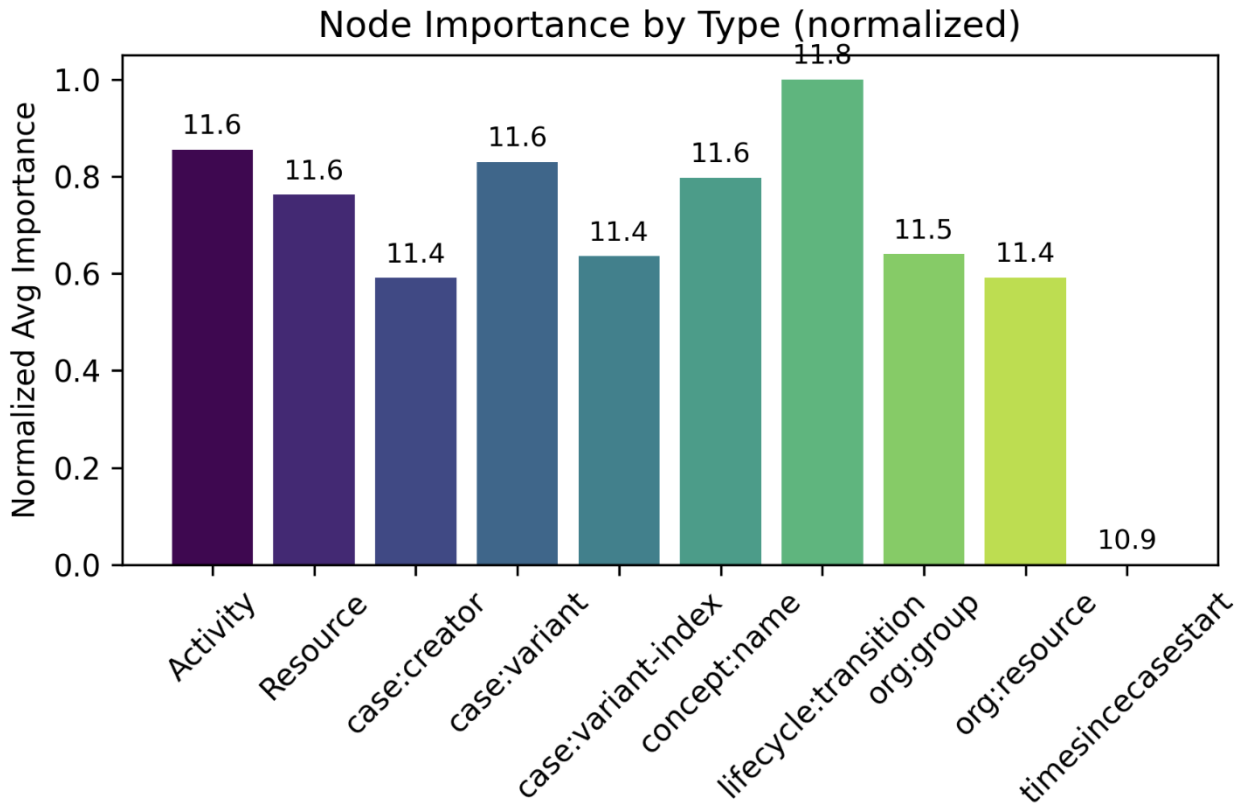


Figure 8: Explanation by type done by TAPGExplainer.

8 Discussion

This research comprised multiple interrelated, labour-intensive stages. The high dimensionality and structural heterogeneity of process data require from each step accurate and specialized methodological procedures from exploratory analysis and sample preparation to variant tuning of architectures and selection of loss functions. Therefore this paper demonstrates that the combination of the heterogeneous graph neural network PROPHET and the type-sensitive explainer TAPGExplainer can simultaneously achieve high predictive accuracy and structurally sound interpretability of predictions in a predictive process monitoring (PPM) task, making the inference process in the explanation optimized for business queries. Firstly, During the training phase, we used both classical recurrent networks and advanced graph neural models; validation protocols were carefully formulated and implemented to ensure reproducibility. From this, the potential of the GNN approach was retrieved, as well as the best model, which was used later.

Secondly, This analysis confirms that XAI for PPM lacks a unified formal framework for comparing explanationary PPM methods. Consequently, a methodological gap

persists between model construction and rigorous explainability assessment. The study concludes that a universal tool for explainable predictive analytics is not yet feasible. As part of our work, we performed an end-to-end comparison of a whole range of explainers (GNNExplainer, PGExplainer, AttentionExplainer, SubGraphX) on several representative process datasets. To improve the objectivity of the evaluation, we introduced four classes of metrics: fidelity, comprehensiveness, time, each revealing the practical utility of the explainer in a different way. In addition, we proposed our own TAPG Explainer method-a multi-step algorithm for finding minimal subgraphs with repeated likelihood checking. On PPM models, it shows a steady-though not radical-increase in the accuracy of interpretations.

Nevertheless, the work has several limitations. First, the analysis only covered the task of predicting the next activity; metrics of remaining time and probabilities of process outcomes were not considered. Second, the study was laboratory in nature: the subjective perception of explanations by domain experts was not measured, and their impact on operational decisions was not studied. Third, the explanations were generated in offline mode; the issues of integration into streaming monitoring with strict requirements for analysis latency were left open, although they were considered. Finally, no methodological way was given to evaluate the prediction model on process logs as originally planned.

The results suggest a number of directions for further development. First, there is a need for a single open benchmark for XAI-methods on graph neural networks, including both synthetic and real industry data with clearly annotated “ground-truth” explanations. Second, the explanability analysis should be complemented by an assessment of its impact on down-stream tasks: robustness to data drift, ability to detect bias, and applicability to real-world business cases. Finally, integrating causal graph models with human-centric metrics could establish the rigorous research programm currently lacking in the XAI community. This paper aims to initiate a systematic research agenda that will narrow the methodological gap and advance transparent, reliable applications of graph neural networks in predictive process monitoring. Equally important is human-centered evaluation. Future work should test the approach on real business cases and evaluate visualisation quality with domain experts. Multi-center UX studies with process analysts will help correlate objective fidelity metrics with subjective criteria of trust and usefulness, as well as identify the amount and format of explanations sufficient for the user to make a decision.

9 Conclusion

This paper covered all the stated objectives:

1. *To conduct a fundamental analysis and comparison of graph neural network (GNN) models from the perspective of accuracy and explainability*.
2. *To propose a method for explainable predictive process monitoring*.
3. *To create a benchmark for future researches in the field of GNN Explainability*.

Although this paper has growth points, it has revealed major insights and revealed more fully the problem of explainability in models for PPM problems. Moreover, a new approach in the explainability of graph neural networks that focuses on process logs was proposed.

References

[1] Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. "evaluating explainability for graph neural networks". *Scientific Data*, 10, 2023.

[2] A. Al-Jebrni, H. Cai, and L. Jiang. Predicting the next process event using convolutional neural networks. In *Proceedings of the IEEE International Conference on Progress in Informatics and Computing*, pages 332–338, 2018.

[3] Alireza Alibakhshi and Erfan Hassannayebi. Towards an enhanced next activity prediction using attention based neural networks. *Cluster Computing*, 28(Article 54), 2025.

[4] Paolo Ceravolo, Sylvio Barbon Junior, Ernesto Damiani, and Wil M. P. van der Aalst. Tuning machine learning to address process mining requirements. *IEEE Access*, 12:24583–24595, February 2024.

[5] Paolo Ceravolo, Marco Comuzzi, Jochen De Weerdt, Chiara Di Francescomarino, and Fabrizio Maria Maggi. Predictive process monitoring: concepts, challenges, and future research directions. *Process Science*, 1(2), 2024.

[6] Juan G. Colonna, Ahmed A. Fares, Márcio Duarte, and Ricardo Sousa. Process mining embeddings: Learning vector representations for petri nets. *arXiv preprint arXiv:2404.17129*, 2024.

[7] Wil Van der Aalst Helen Schonenberg Minseok Song. Time prediction based on process mining. *Information Systems*, 2011.

[8] Sebastiano Dissegna and Chiara Di Francescomarino. Graph neural networks for ppm: Review and benchmark for next activity predictions. In Katarzyna Gdowska, María Teresa Gómez-López, and Jana-Rebecca Rehse, editors, *Business Process Management Workshops. BPM 2024. Lecture Notes in Business Information Processing*, volume 534 of *Lecture Notes in Business Information Processing*, pages 31–43. Springer, Cham, 2025.

[9] Le Toan Duong, Louise Travé-Massuyès, Audine Subias, and Christophe Merle. Remaining cycle time prediction with graph neural networks for predictive process monitoring. In *Proceedings of the Association for Computing Machinery*. Association for Computing Machinery, 2023.

[10] Manuel Lama Efrén Rama-Maneiro, Juan C. Vidal. Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Transactions on Services Computing*, 2021.

[11] J. Evermann, J.-R. Rehse, and P. Fettke. Predicting process behaviour using deep learning. *Decision Support Systems*, 100:129–140, 2017.

[12] Marlon Dumas Chiara Ghidini Fabrizio Maria Maggi, Chiara Di Francescomarino. Predictive monitoring of business processes. *Advanced Information Systems Engineering*, 2014.

[13] C. D. Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, and A. Yeshchenko. An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring. In *Proceedings of the 15th International Conference on Business Process Management*, pages 252–268, 2017.

[14] Riccardo Galanti, Bernat Coma-Puig, Massimiliano de Leoni, Josep Carmona, and Nicolò Navarin. Explainable predictive process monitoring. *arXiv preprint arXiv:2008.01807*, 2020.

[15] Maximilian Harl, Sven Weinzierl, Mathias Stierle, and Martin Matzner. Explainable predictive business process monitoring using gated graph neural networks. *Journal of Decision Systems*, 29(sup1):312–327, 2020.

[16] Markku Hinkka, Teemu Lehto, and Keijo Heljanko. Exploiting event log event attributes in rnn based prediction. In *Data-Driven Process Discovery and Analysis*, Lecture Notes in Business Information Processing, pages 67–85. Springer International Publishing, 2020.

[17] Sylvio Barbon Jr., Paolo Ceravolo, Rafael S. Oyamada, and Gabriel M. Tavares. Trace encoding in process mining: A survey and benchmarking. *Engineering Applications of Artificial Intelligence*, 126:107028, 2023.

[18] A. Khan et al. Memory-augmented neural networks for predictive process analytics. *CoRR*, abs/1802.00938, 2018.

[19] Sungkyu Kim, Marco Comuzzi, and Chiara Di Francescomarino. Explaining the impact of design choices on model quality in predictive process monitoring. *Journal of Intelligent Information Systems*, 2024.

[20] L. Lin, L. Wen, and J. Wang. Mm-pred: A deep predictive model for multi-attribute event sequence. In *Proceedings of the SIAM International Conference on Data Mining*, pages 118–126, 2019.

[21] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774. Curran Associates, Inc., 2017.

[22] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. ”parameterized explainer for graph neural network”. In *Advances in Neural Information Processing Systems*, volume 33, 2021.

[23] N. Mehdiyev, J. Evermann, and P. Fettke. A novel business process prediction model using a deep learning method. *Business & Information Systems Engineering*, 62:143–157, 2018.

[24] A. Metzger and A. Neubauer. Considering non-sequential control flows for process prediction with recurrent neural networks. In *Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications*, pages 268–272, 2018.

[25] N. Navarin, B. Vincenzi, M. Polato, and A. Sperduti. Lstm networks for data-aware remaining time prediction of business process instances. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1–7, 2017.

[26] Vincenzo Pasquadibisceglie, Annalisa Appice, and Donato Malerba. Lupin: A llm approach for activity suffix prediction in business process event logs. In *Proceedings of the 6th International Conference on Process Mining (ICPM 2024)*, pages 1–8. IEEE, 2024.

[27] Vincenzo Pasquadibisceglie, Raffaele Scaringi, Annalisa Appice, Giovanna Castellano, and Donato Malerba. Prophet: Explainable predictive process monitoring with heterogeneous graph neural networks. *IEEE Transactions on Services Computing*, 17(6):4111–4124, 2025.

[28] Patrick Philipp, Ruben Jacob, Sebastian Robert, and Jürgen Beyerer. Predictive analysis of business processes using neural networks with attention mechanism. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 225–230. IEEE, 2020.

[29] Efren Rama-Maneiro, Juan C. Vidal, and Manuel Lama. Embedding graph convolutional networks in recurrent neural networks for predictive monitoring. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[30] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[31] Williams Rizzi, Marco Comuzzi, Chiara Di Francescomarino, Chiara Ghidini, Suhwan Lee, Fabrizio Maria Maggi, and Alexander Nolte. Explainable predictive process monitoring: a user evaluation. *Process Science*, 1, 2024.

[32] Williams Rizzi, Chiara Di Francescomarino, and Fabrizio Maria Maggi. Explainability in predictive process monitoring: When understanding helps improving. In *Business Process Management Forum. BPM 2020*, volume 392 of *Lecture Notes in Business Information Processing*, pages 141–158. Springer, Cham, 2020.

[33] Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[34] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks. In Eric Dubois and

Klaus Pohl, editors, *Advanced Information Systems Engineering – CAiSE 2017, Lecture Notes in Computer Science*, volume 10253 of *Lecture Notes in Computer Science*, pages 477–492. Springer, Cham, 2017.

[35] Wil M. P. van der Aalst and Josep Carmona, editors. *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*. Springer Cham, Cham, Switzerland, 1 edition, 2022.

[36] W.M.P. van der Aalst. Process mining: Discovery, conformance and enhancement of business processes. *Springer, Heidelberg*, 2011.

[37] W.M.P. van der Aalst. Process mining: Data science in action. *Springer, Heidelberg*, 2016.

[38] Bemali Wickramanayake, Zhipeng He, Chun Ouyang, Catarina Moreira, Yue Xu, and Renuka Sindhgatta. Building interpretable models for business process prediction using shared and specialised attention mechanisms. *Knowledge-Based Systems*, 248:108773, 2022.

[39] Junyi Xu and Xianwen Fang. Att-blkan: A hybrid deep learning model combining attention is used to enhance business process prediction. *IEEE Access*, 13:36175–36189, 2025.

[40] Yifei Xu and Huan FANG. Ppmllm embedding and aligning process data in large language models for predictive process monitoring. *SSRN Electronic Journal*, pages 1–25, 2024. Posted: 16 December 2024.

[41] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. ”gnnexplainer: Generating explanations for graph neural networks”. In *Advances in Neural Information Processing Systems*, volume 32, pages 9240–9251, 2019.

[42] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. ”on explainability of graph neural networks via subgraph explorations”. In Marina Meilă and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12241–12252. PMLR, 2021.