Detecting Anomalies in Healthcare Processes: A K-NN Graph-Based approach

Iuliana Malina Grigore[1,*,†], Azin Moradbeikie[1,†] and Sylvio Barbon Junior[1,†]

[1]Università degli Studi di Trieste, Piazzale Europa, 1, 34127 Trieste TS, Italy

Abstract

Detecting anomalies in healthcare processes helps identify irregular patterns that may point to medical errors, inefficiencies, or departures from clinical protocols. As event logs become more common in clinical systems, they offer a valuable resource for understanding and improving how healthcare is delivered. This increases the relevance of process mining, which supports the analysis and improvement of real-world workflows by examining the sequence of events recorded in these logs. However, working with real event data can be difficult due to its complexity and variability. To manage this, we use graph-based models that highlight the relationships between activities without generating overly complicated, hard-to-read process maps. In this study, we focus on K-Nearest Neighbors (K-NN) graph modeling, which does not require setting the number of clusters in advance. This flexibility allows for a more detailed view of the process landscape and helps identify subtle variations or unusual behaviors. Using a hospital billing dataset, we show that graph-based methods—especially K-NN—can better group typical and unusual patterns when compared to more traditional techniques. These results support the use of graph models as a practical tool for reviewing and improving healthcare processes, with the potential to support better management and patient outcomes.

1. Introduction

In healthcare information systems, anomalies in process executions reflect atypical patterns related to patients, medical conditions, or healthcare operations. Detecting these anomalies is important for triggering early warnings, enabling timely interventions, supporting informed decision-making, and identifying opportunities for process improvement. Anomaly detection also plays a key role in uncovering potential fraud and anticipating emerging clinical issues. However, the inherently complex and sensitive nature of healthcare data makes anomaly detection particularly challenging.

Process mining (PM) has become a valuable approach for analysing and improving clinical workflows using event data of information systems. However, its application in healthcare is complicated by the high variability and unpredictability of treatment pathways, which often differ depending on a patient's condition and treatment plan [1]. This variability, combined with the fragmented and voluminous data stored in electronic health records and other systems, creates significant barriers to identifying inefficiencies and process bottlenecks.

Unlike traditional analytics methods, which are often unable to capture the dynamic and non-linear characteristics of healthcare, process mining provides more accurate and transparent representations of real-world workflows [2]. It supports both process model discovery and performance evaluation while being insensitive to noise and incomplete data that are common in healthcare. By uncovering hidden patterns, deviations, and inefficiencies, process mining enables healthcare organisations to make data-driven decisions that improve service delivery [2].

These capabilities ultimately support the fourfold goal of improving healthcare [3]: (i) improving population health through better analysis of care pathways; (ii) improving the patient experience by optimising service delivery; (iii) reducing costs by eliminating inefficiencies; and (iv) supporting medical

Ital-IA 2025: 5th National Conference on Artificial Intelligence, organized by CINI, June 23-24, 2025, Trieste, Italy

*Corresponding author.

†These authors contributed equally.

iulianamalina.grigore@phd.units.it (I. M. Grigore); azin.moradbeikie@dia.units.it (A. Moradbeikie); sylvio.barbonjunior@units.it (S. B. Junior)

0009-0007-4602-2714 (I. M. Grigore); 0000-0002-6374-7780 (A. Moradbeikie); 0000-0002-4988-0702 (S. B. Junior)

staff by analysing workload distribution and resource utilisation [3, 4]. In recent years, process mining has been applied in a wide range of healthcare use cases, as demonstrated by several comprehensive literature reviews [5, 2, 6, 7, 8]. Among the techniques, trace clustering has attracted attention due to its ability to group similar process executions and uncover meaningful behavioural patterns [9]. This is particularly important for variant analysis, which aims to understand and compare different execution paths within the same process. For example, in [10], trace clustering was used to identify dominant behavioural patterns in healthcare processes, providing a simplified yet insightful representation of healthcare workflows. These patterns supported a clearer understanding of typical workflows and enabled the analysis of process variations for better clinical decision making.

However, as outlined in [11], the unique characteristics of healthcare processes — namely their unstructured and highly variable nature — limit the effectiveness of traditional process mining approaches. These methods often result in overly complex, spaghetti-like models that are difficult to interpret. Although techniques such as trace clustering and semantic aggregation can reduce some of the complexity, they can also obscure rare but clinically significant behaviors, limiting their applicability in practice.

To overcome these challenges, machine learning is increasingly using graph-based techniques that are well-suited to capturing the relational complexity of real-life event logs. Approaches such as Graph Neural Networks (GNNs) [12], heterogeneous networks [13], and cross-institutional graphs [14] leverage the structure of the data to enable accurate anomaly detection, system-level insights, and better interpretability — even in noisy and highly varying environments.

Building on the strengths of process mining and machine learning, we propose a graph-based approach that uses k-nearest neighbours (k-NN) to build a process graph where the nodes represent individual traces and the edges encode the similarity of behaviour. This hybrid method combines the precision of extracting behavioural features from process mining with the modelling flexibility of graph-based learning to address challenges such as concurrency and anomaly detection in hospital billing data. By leveraging the topology of the graph, our approach improves the interpretability of process variations and provides actionable insights in complex and variable healthcare environments. The main contributions of this work are:

- Evaluate the effectiveness of graph-based modeling for real-world anomaly detection.
- Interpret anomalies within the healthcare domain.

The remainder of this paper is organized as follows: Section 2 outlines the proposed approach, the dataset, and experimental results. Section 3 concludes the paper and outlines directions for future work.

2. Experiments, Results and Discussions

This section describes the data set used in the experiments and the associated parameter settings. The experimental methodology follows the approach described in [15], which uses a k-nearest neighbors (K-NN) graph to capture competing behaviors between events and identify anomalies within process traces. In this framework, each single trace is represented as a node in the graph, and similarity is evaluated from multiple perspectives, including activity, resource, and transition features.

To achieve this, the traces are first encoded (i.e., profiled) into numeric vectors using one-hot encoding [16]. Edges in the graph are created based on Levenshtein distance to link similar traces. This graph structure supports community detection through modularity optimization and enables the identification of clusters of behaviorally similar trace variants. Variants that deviate significantly from the patterns of their associated community are labeled as anomalies.

2.1. Hospital Billing Event Log

We used the Hospital Billing event log[1] to evaluate our proposed approach. This event log comes from the financial modules of the ERP system of a regional hospital and was collected and published

[1] https://data.4tu.nl/articles/dataset/Hospital_Billing_-_Event_Log/12705113/1

by Eindhoven University of Technology as part of their process mining research. The log consists of 451,359 events distributed over 100,000 unique traces, corresponding to the final stages of billing-related transactions. In the context of process mining, each trace represents a sequence of recorded events corresponding to a single execution of a process, for example, all activities related to a specific patient billing case. Each event contains 23 attributes, such as activity name, timestamp, resource, and case identifier, and the log records 18 unique activities (new, fin, release, coke ok, billed, delete, reopen, change diagn, storno, reject, code nok, set status, chang end, manual, join-pat, code error, sdc_behan, and empty). On average, each trace contains five events, with the longest trace having 217 events and the shortest containing only a single event. Given the size and complexity of the log, we performed the usual pre-processing steps, including sorting by timestamp and filtering out irrelevant attributes.

2.2. Implementation

The sequence of events associated with a particular patient was grouped into a trace, representing a single instance of the billing process. These traces were then profiled along three dimensions (activity, resource, and transition) to support their transformation into a graph-based representation. Similarity between traces was computed using Levenshtein distance applied separately to each of the three profile types: (i) activity profiles, represented as binary vectors indicating the presence (1) or absence (0) of each activity; (ii) transition patterns, reflecting observed pairs of consecutive activities; and (iii) resource usage, represented as count vectors for each resource involved in the trace. For all experiments, we set $k = 3$ as the default number of neighbors in the k-NN graph. In the subsequent step, we applied the Louvain method for community detection, using a resolution parameter of 1.0.

The approach was implemented as a Python prototype, which is publicly available in the project repository and can be easily integrated[2]. The implementation uses *pm4py*[3] for event log pre-processing and object-centric profiling, while *NetworkX* is used for process graph creation and community detection.

2.3. Analysis

We split our analysis into chunks by selecting a 10% sample of the data each time to examine seasonality and potential short-term patterns. We performed 10 independent runs, each with a different 10% subset and a unique random initialization, to account for the non-deterministic nature of the method. This approach allowed us to assess the effects of initialization on community formation and served as a form of cross-validation to assess the stability and variability of the results.

Table 1 shows the results of 10 runs with different random graph generation processes, showing consistent anomaly detection rates (about 2.5%) but some variation in the number and structure of communities. Among these runs, we focused on identifying a case that showed some internal variation in detected variants within the same community. The most promising candidates were the runs 10, 3, and 2. Since the latter two runs gave identical community structure and anomaly detection results, we selected the Run 3 for further analysis.

Table 2 shows the distribution and characteristics of the detected communities when analyzing the cases generated with Run 3. These communities differ significantly in terms of size, internal similarity, and behavioral diversity, which is reflected in the number of unique variants. The majority of cases (approximately 97%) are concentrated in three large communities— communities 0, 1, and 2 — all characterized by a similarity score of 1 and classified as normal. In contrast, the remaining 3% of cases are spread across six smaller communities, all of which are labeled as anomalies.

A closer examination of the anomalous communities reveals distinct behavioral patterns. Most of these smaller communities consist of only one or two cases, indicating very unique or isolated behavior. Community 8, for example, is characterized by a self-loop in the activity `set status` and the presence of a previously unseen activity, `change end`. This activity also occurs in communities 4 and 5, indicating a strong association with anomalous process paths.

[2] https://github.com/azinmoradbeikie/Ital-IA_2025_Variant_Anomalies_Healthcare
[3] https://github.com/pm4py/pm4py-core

Table 1
Community characteristics across 10 runs with different random initializations with a total of 10,000 cases.

# Run	# Communities	# Anomalous Communities	# of Detected Anomaly Cases
1	10	7	248 (2.5% of cases)
2	9	6	252 (2.5% of cases)
3	9	6	253 (2.5% of cases)
4	11	8	240 (2.4% of cases)
5	9	6	209 (2.1% of cases)
6	8	5	264 (2.6% of cases)
7	8	5	244 (2.4% of cases)
8	6	3	249 (2.5% of cases)
9	8	5	256 (2.5% of cases)
10	11	8	243 (2.4% of cases)

Table 2
Anomalies and number (#) of variants across communities using the Run 3.

Community	Anomaly	# Variants	# Case
0	*None*	104	4056
1	*None*	68	2433
2	*None*	43	3258
3	Anomaly	25	247
4	Anomaly	1	1
5	Anomaly	2	2
6	Anomaly	1	1
7	Anomaly	1	1
8	Anomaly	1	1

Community 4 contains two similar variants with the sequence `fin` → `release` → `code ok` → `billed`, with one variant repeating the entire sequence. The repetition is interrupted by `storno` → `reject`, which signals a deviation in the process flow. Community 7 has a similar pattern, followed by an additional fragment: `storno` → `set status`, which adds to the complexity.

Community 6 has structural similarities, but replaces `code ok` with `code nok` and introduces the activities `reopen` and `delete`. Community 5 shows three repetitions of the sequence `fin` → `release` → `change end`, with a fourth attempt ending with `billed` instead of `change end`, which again signals a deviation from the standard behavior.

Community 3 is still labeled anomalous, but differs from the other small communities in terms of size: it includes a significantly higher number of cases and variants. Most traces in this community start with `new` → `change diagn` and end with `delete`, although some end with `set status` or remain incomplete (`empty`). In the middle of these traces there are often reprocessing loops such as `fin` → `release` → `reopen`, `fin` → `release` → `code ok` → `reopen` and `fin` → `release` → `code nok`, indicating attempts to re-evaluate or revise decisions.

On the other hand, normal communities also show a wide range of behaviors, which is reflected in their high number of variants. Community 2 typically begins with `new` → `fin`, although some exceptions include traces ending with `delete`, or containing only `new`, or repeatedly reverting to `new`. In Community 1, most cases end with `billed`, but a few end with `join-pat`, `set status`, or even `code ok`. In the middle of traces, there are self-loops with `manual` and `code nok` and repeated sequences like `code nok` → `billed` → `storno` → `reject` or similar sequences with `code ok` instead of `code nok`.

Community 2 also includes loops with `join-pat` and repeated patterns such as `release` → `code nok` followed by `fin`, `delete`, or `reopen`. Community 0 usually starts with `new` → `change diagn` and ends with various activities such as `billed`, `code ok`, `manual`, or `set status`. This community

exhibits self-loops at `change diagn`, `fin`, and `manual` and shares many of the repetitive patterns observed in communities 1 and 2. In particular, the sequence `fin` $\rightarrow$ `release` $\rightarrow$ `reopen` $\rightarrow$ `fin` is found in all three normal communities and also appears in community 3.

This analysis explains why larger communities often contain more variants. Even if these variants follow different paths, they generally show similar behavior and lead to the same results. This consistency helps to avoid misclassification, as the clustering approach distinguishes cases that differ in form but not in function.

Communities are classified as normal or abnormal based on the frequency of their variants. If the community had 3 more cases, which would increase the frequency of its variants, it would probably be classified as normal. Anomalous communities, on the other hand, are characterized by rare behaviors that occur only a few times in the dataset.

A closer inspection of the anomalous communities reveals behavioral patterns that may reflect inefficiencies, exceptional cases, or irregular healthcare system usage. Many of these anomalous behaviours suggest that the change end may correspond to undocumented or legacy process steps. Other patterns point to repeated and potentially unnecessary attempts to complete a billing process, which may reflect unclear feedback mechanisms or insufficient validations in the system interface. Finally, the self-loops indicate cases where users may be struggling to resolve process errors, leading to repeated corrections or cancellations. Such patterns highlight potential weaknesses in workflow support, especially in scenarios requiring manual intervention.

The results indicate that many anomalies could be linked to unclear process definitions, inconsistent error handling, or insufficient system guidance. These findings provide actionable feedback for healthcare information system administrators, who may consider auditing rare activities, improving system feedback, and enhancing support for correct process execution to reduce the emergence of anomalous behavior.

While the proposed approach demonstrates promising results in identifying anomalous behavior within healthcare processes, it presents several limitations that must be considered. The computational cost of constructing the k-NN graph grows significantly with the number of traces. The use of Levenshtein distance for measuring similarity between traces is especially expensive, as it requires pairwise comparisons that scale quadratically with the number of cases. This limitation affects both runtime and memory usage, making the approach less practical for very large datasets. Moreover, the method is sensitive to key parameters, in particular the number of neighbors k and the distance threshold used to construct the graph. Lastly, interpretability and reproducibility are affected by the stochastic nature of the community detection algorithm and the graph construction process.

3. Conclusion and Future Work

The approach highlights the value of combining process mining with graph-based techniques such as k-nearest neighbors (k-NN) to improve the detection of anomalies in healthcare workflows. The analysis shows that larger communities exhibit consistent and typical process behavior, while smaller, anomalous communities reflect unique and rare deviations that stand out in the data. By examining these anomalies in detail, we can uncover hidden inefficiencies and identify potential problems in clinical processes. This distinction not only helps to identify deviations from standard behavior but also helps to refine healthcare operations by focusing on outliers that may lead to operational errors or missed opportunities for improvement. This hybrid approach provides more accurate and actionable insights into complex healthcare environments and supports better decision-making and process optimization.

Acknowledgments

This work has been supported by the European Union – NextGenerationEU under the NRRP (M4C2I1.1), REPA (aRtificial intElligence for Process Analytics) project, and by the PORTRAIT project (Port to Rail

Digital Twin in the Adriatic Region), funded by the PR FESR 2021–2027, Action A1.1.2, DGR 784/2023 of the FVG Region (Italy).

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT and Grammarly in order to: Grammar and spelling check, Paraphrase, and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] Álvaro Rebuge, D. R. Ferreira, Business process analysis in healthcare environments: A methodology based on process mining, Information Systems 37 (2012) 99–116. Management and Engineering of Process-Aware Information Systems.

[2] E. Rojas, J. Munoz-Gama, M. Sepúlveda, D. Capurro, Process mining in healthcare: A literature review, Journal of Biomedical Informatics 61 (2016) 224–236. doi:https://doi.org/10.1016/j.jbi.2016.04.007.

[3] C. S. T. Bodenheimer, From triple to quadruple aim: care of the patient requires care of the provider, The Annals of Family Medicine: 12 (6), 2014.

[4] J. W. D.M. Berwick, T.W. Nolan, The triple aim: care, health, and cost, Health Aff. 27 (3) (2008).

[5] T. G. Erdogan, A. Tarhan, Systematic mapping of process mining studies in healthcare, IEEE Access 6 (2018) 24543–24567.

[6] E. Batista, A. Solanas, Process mining in healthcare: A systematic review, in: 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA), 2018, pp. 1–6.

[7] A. R. Guzzo, Antonella, E. Vocaturo, Process mining applications in the healthcare domain: A comprehensive review, WIREs Data Mining and Knowledge Discovery 12, fasc. 2 (2022).

[8] L. Aversano, M. Iammarino, A. Madau, G. Pirlo, G. Semeraro, Process mining applications in healthcare: A systematic literature review, PeerJ Computer Science 11 (2025).

[9] R. P. J. C. Bose, W. M. P. van der Aalst, Trace clustering based on conserved patterns: Towards achieving better process models, in: S. Rinderle-Ma, S. Sadiq, F. Leymann (Eds.), Business Process Management Workshops, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 170–181.

[10] M. Faizan, M. F. Zuhairi, S. Ismail, Process discovery enhancement with trace clustering and profilin, Annals of Emerging Technologies in Computing 5, fasc. 4 (2021).

[11] J. Munoz-Gama, N. Martin, C. Fernandez-Llatas, O. A. Johnson, M. Sepúlveda, E. Helm, V. Galvez-Yanjari, E. Rojas, A. Martinez-Millana, D. Aloini, I. A. Amantea, R. Andrews, M. Arias, I. Beerepoot, E. Benevento, A. Burattin, D. Capurro, J. Carmona, M. Comuzzi, B. Dalmas, R. de la Fuente, C. Di Francescomarino, C. Di Ciccio, R. Gatta, C. Ghidini, F. Gonzalez-Lopez, G. Ibanez-Sanchez, H. B. Klasky, A. Prima Kurniati, X. Lu, F. Mannhardt, R. Mans, M. Marcos, R. Medeiros de Carvalho, M. Pegoraro, S. K. Poon, L. Pufahl, H. A. Reijers, S. Remy, S. Rinderle-Ma, L. Sacchi, F. Seoane, M. Song, A. Stefanini, E. Sulis, A. H. ter Hofstede, P. J. Toussaint, V. Traver, Z. Valero-Ramon, I. van de Weerd, W. M. van der Aalst, R. Vanwersch, M. Weske, M. T. Wynn, F. Zerbato, Process mining for healthcare: Characteristics and challenges, Journal of Biomedical Informatics 127 (2022) 103994.

[12] P. Kisanga, I. Woungang, I. Traoré, G. H. S. Carvalho, Network anomaly detection using a graph neural network, 2023 International Conference on Computing, Networking and Communications (ICNC) (2023) 61–65.

[13] J. Liu, E. Bier, A. Wilson, J. A. Guerra-Gomez, T. Honda, K. Sricharan, L. Gilpin, D. Davies, Graph analysis for detecting fraud, waste, and abuse in health-care data, AI Mag. 37 (2016) 33–46. doi:10.1609/aimag.v37i2.2630.

[14] N. H, O. OA, L. MA, O. M. Grady SK, O. O, K. HB, L. A, N. J. Ward M, Anomaly detection in

electronic health records across hospital networks: Integrating machine learning with graph algorithms, IEEE J Biomed Health Inform (2025).

[15] I. M. Grigore, G. M. Tavares, S. B. Junior, Beyond flattening: Detecting concurrency anomalies using k-nn graph-based modeling in object-centric event logs, in: R. M. Czekster, P. Milazzo (Eds.), From Data to Models and Back, Springer Nature Switzerland, Cham, 2025, pp. 116–136.

[16] G. M. Tavares, R. S. Oyamada, S. B. Junior, P. Ceravolo, Trace encoding in process mining: A survey and benchmarking, Engineering Applications of Artificial Intelligence 126 (2023) 107028.