escola superior de tecnologia e gestão
instituto politécnico de leiria

CASE ID DETECTION IN UNLABELLED EVENT LOGS FOR PROCESS MINING

ANDRÉ ALEXANDRE DOS SANTOS VICENTE

Leiria, September of 2023

escola superior de tecnologia e gestão
instituto politécnico de leiria

CASE ID DETECTION IN UNLABELLED EVENT LOGS FOR PROCESS MINING

ANDRÉ ALEXANDRE DOS SANTOS VICENTE

Master's thesis project carried out under the supervision of Professors Rui Pedro Charters Lopes Rijo, Ricardo Filipe Gonçalves Martinho, and Carlos Fernando de Almeida Grilo, faculty members of the School of Technology and Management at the Polytechnic University of Leiria.

Leiria, September of 2023

ACKNOWLEDGEMENTS

I extend my heartfelt gratitude to all who have played a pivotal role in the successful completion of my master's project. This journey has been both a challenge and a source of great reward, and it is with immense appreciation that I acknowledge the contributions of numerous individuals.

Foremost, I express my profound thanks to my academic advisors and co-advisors, Professor Rui Pedro Charters Lopes Rijo, Professor Ricardo Filipes Gonçalves Martinho, and Professor Carlos Fernando de Almeida Grilo. Their unwavering guidance, profound insights, and constant encouragement were instrumental in shaping the direction of this work.

To my family, particularly my mother, Célia Santos, I owe an immeasurable debt of gratitude. Your unwavering love, steadfast support, and enduring belief in my capabilities have served as the bedrock and driving force behind my academic journey.

I extend my gratitude to the countless others who have contributed in diverse ways, collectively forming the support network that made this significant achievement possible. Your involvement has been invaluable.

Thank you all for being integral to this milestone accomplishment.

ABSTRACT

In the realm of data science, event logs serve as valuable sources of information, capturing sequences of events or activities in various processes. However, when dealing with unlabelled event logs, the absence of a designated Case ID column poses a critical challenge, hindering the understanding of relationships and dependencies among events within a case or process.

Motivated by the increasing adoption of data-driven decision-making and the need for efficient data analysis techniques, this master's project presents the "Case ID Column Identification Library" project. This library aims to streamline data preprocessing and enhance the efficiency of subsequent data analysis tasks by automatically identifying the Case ID column in unlabelled event logs.

The project's objective is to develop a versatile and user-friendly library that incorporates multiple methods, including a Convolutional Neural Network (CNN) and a parameterizable heuristic approach, to accurately identify the Case ID column. By offering flexibility to users, they can choose individual methods or a combination of methods based on their specific requirements, along with adjusting heuristic-based formula coefficients and settings for fine-tuning the identification process.

This report presents a comprehensive exploration of related work, methodology, data understanding, methods for Case ID column identification, software library development, and experimental results. The results demonstrate the effectiveness of the proposed methods and their implications for decision support systems.

Keywords: Process Mining, CNN, Case ID Identification, Attribute Identification

RESUMO

No âmbito da ciência de dados, os registos de eventos servem como fontes valiosas de informação, capturando sequências de eventos ou atividades em vários processos. No entanto, ao lidar com registos de eventos não rotulados, a ausência de uma coluna designada de Case ID representa um desafio crítico, dificultando a compreensão das relações e dependências entre eventos dentro de um caso ou processo.

Motivado pela crescente adoção da tomada de decisão baseada em dados e pela necessidade de técnicas eficientes de análise de dados, este projeto de mestrado apresenta o projeto "Biblioteca de Identificação da coluna Case ID". Esta biblioteca tem como objetivo simplificar o pré-processamento de dados e melhorar a eficiência das tarefas subsequentes de análise de dados, identificando automaticamente a coluna em registos de eventos não rotulados.

O objetivo do projeto é desenvolver uma biblioteca versátil e de fácil utilização que incorpore múltiplos métodos, incluindo uma Rede Neural Convolucional e uma abordagem heurística parametrizável, para identificar com precisão a coluna Case ID. Ao oferecer flexibilidade aos utilizadores, eles podem escolher métodos individuais ou uma combinação de métodos com base nos seus requisitos específicos, além de ajustar os coeficientes de fórmulas baseadas em heurísticas e configurações para ajustar o processo de identificação.

Este relatório apresenta uma exploração abrangente do trabalho relacionado, metodologia, compreensão dos dados, métodos para identificação da coluna Case ID, desenvolvimento da biblioteca de software e resultados experimentais. Os resultados demonstram a eficácia dos métodos propostos e as suas implicações para sistemas de apoio à decisão.

Palavras-chave: Process Mining, CNN, Identificação de Case ID, Identificação de Atributos

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI	Artificial Intelligence.
ANNs	Artificial Neural Networks.
AUC-ROC	Area Under the Receiver Operating Characteristic Curve.
BPI	Business Process Intelligence.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
CSV	Comma-Separated Values.
DSS	Decision Support System.
EDE	Event Density Embedding.
GPU	Graphics Processing Unit.
IEEE	Institute of Electrical and Electronics Engineers.
MCC	Matthews Correlation Coefficient.
PM	Process Mining.
SNA	Social Network Analysis.
SNNs	Simulated Neural Networks.

XES	eXtensible Event Stream.

1 INTRODUCTION

The development of advanced data analysis techniques and the increasing adoption of data-driven decision-making have significantly impacted various domains, from business and healthcare to engineering and research. In this context, event logs have emerged as valuable sources of information, capturing sequences of events or activities in various business processes, within organizations.

Event logs consist of structured records of events or activities in organizational processes, documenting actions chronologically with timestamps and relevant data such as user actions. Process Mining (PM), a set of techniques and tools, is used to extract insights, monitor, and optimize real processes by analysing event logs. It plays a crucial role in understanding process efficiency and potential improvements. PM relies on event logs to reconstruct and analyse processes effectively.

Event logs are extensively used for process analysis, optimization, and anomaly detection. However, when dealing with unlabelled event logs, a critical challenge arises: the absence of a designated Case ID column. The Case ID column is crucial for understanding the relationships and dependencies among events within a case or process.

In the realm of Data Science, researchers and analysts often face the arduous task of manually labelling event logs, which can be time-consuming and prone to errors, especially for large datasets [1]. Additionally, the lack of a Case ID column restricts the effective application of PM techniques, limiting the ability to gain valuable insights and conduct comprehensive analyses.

The objective of this work is to develop a versatile and user-friendly library for labelling unlabelled event logs, often used to record sequences of events of business processes in various domains. These logs often lack a designated identifier, such as a Case ID, essential for further analysis and insights. Emphasizing the importance of data preprocessing and automated tools in facilitating data analysis, the proposed library aims to contribute to ongoing research in PM, data analytics, and data-driven decision-making. Additionally, it is a component of the Prom4Prod project, co-funded by the Portugal 2020 program, aimed at developing a comprehensive PM-based decision support and production management system for various industry sectors. By automating Case ID identification and streamlining the data preparation process, the proposed library becomes an important tool to foster PM analysis and visualizations, and thus facilitating decision-making within these industries.

The motivation behind this work stems from the need to overcome limitations posed by unlabelled event logs, i.e., event logs which do not present, upfront, a clear identification of which column is the designated Case ID of a certain business process. By developing a library that offers multiple methods for Case ID identification, we aim to alleviate manual efforts required for preprocessing data and accelerate the data analysis process. The library aims to empower users from diverse domains, including Data Science, Data Engineering, Business Process Management, to efficiently preprocess their event log data and conduct PM, derive valuable insights, identify bottlenecks, optimize workflows, and improve decision-making. By automating Case ID identification, this library not only reduces the burden of manual labour but also enhances the reliability and accuracy of subsequent PM-related analyses.

Part of the motivation of this project aligns with the growing demand for data-driven solutions, allowing data scientists, analysts, and practitioners to efficiently extract valuable insights from event log data. It serves as a resource for professionals across diverse domains aiming to optimize processes, pinpoint bottlenecks, and instigate data-driven improvements.

This library should aim to offer flexibility to developers, enabling them to select and apply individual methods or combinations based on their specific requirements. Developers should be able to adjust coefficients and settings of the expression to fine-tune the identification process.

With the library's support, developers should also streamline their data preparation workflow, saving valuable time and resources. Additionally, automated Case ID identification enhances the quality and reliability of subsequent analyses, fostering data-driven decision-making and research across fields.

Ultimately, our goal is to contribute to the broader data science community by offering an accessible and robust tool to developers, serving as a decision support system for efficiently and accurately labelling unlabelled event logs.

The rest of the document is structured as follows:

- Chapter 2 - Background and Related Work: In this chapter, we review the existing literature and research related to Case ID column identification in event logs. We explore various methods and approaches used by other researchers in this field, identifying gaps and potential areas for improvement.

- Chapter 3 - Methodology: The methodology chapter outlines the overall approach taken in the project. It describes the steps and procedures used to achieve the project's goals, including data collection, preprocessing, model development, and evaluation.
- Chapter 4 - Data Collection, Understanding and Preparation: In this chapter, we explore the initial steps of the project, focusing on data collection, understanding, and preparation. We explain how we processed and prepared the event logs for further analysis, emphasizing the importance of data quality and completeness.
- Chapter 5 - Convolutional Neural Network Model: This chapter delves into the Convolutional Neural Network (CNN) method used for Case ID column identification. We provide insights into the architecture and training process of the CNN, showcasing its role as one of the key methods in our library.
- Chapter 6 - A Heuristic for Case ID Column Identification: Here, we explore the user-configurable expressions utilized for Case ID column identification. We describe the parameters involved and their impact on the identification process, highlighting their flexibility and adaptability.
- Chapter 7 - Software Library Development: Here, we detail the process of developing the software library that incorporates the case ID identification methods. We discuss the architecture, implementation, and functionalities of the library.
- Chapter 8 - Conclusion and Future Work: The conclusion chapter summarizes the key findings of the project and provides insights into the effectiveness of the proposed methods. It discusses the limitations of the project and potential areas for future research. Additionally, we highlight the project's implications for decision support systems and its contribution to the broader field of data science.

2 BACKGROUND AND RELATED WORK

This chapter aims to provide the necessary background information to contextualize the undertaken research. The chapter begins by introducing the main concepts surrounding this work. Then, the current state of knowledge in this area is described, highlighting main research findings that form the foundation of the work. Next, the research gaps and challenges that this study aims to address are discussed, as well as the potential contributions to the corresponding research field.

2.1 CONCEPTS AND FUNDAMENTALS

In this section, we delve into fundamental concepts at the core of PM. Our exploration is divided into three critical subsections: Process Mining, Event Log, and Process Mining Tools. These sections provide foundational insights for process analysis.

2.1.1 *Process mining*

PM is a research field that stays somewhere between data mining [2] and process modelling [3] and analysis [4], that aims to improve operational business processes through the use of event data [5]. Performing PM starts with the extraction of data from databases in the form of events logs. These event logs then serve as input for the PM algorithms, resulting in analysed data. Finally, the outcomes can be presented in a variety of forms, including Directly-Follows Graphs (DFG) and Petri nets, time-based dotted charts, activity histograms, case tracing and Social Network Analysis (SNA) diagrams (Figure 1).

Figure 1 presents a portrait of the real world, where human interactions take place within an environment. Information systems capture these interactions, giving rise to unlabelled event logs. These event logs serve as the foundation for the PM process. They flow through a sequence involving PM algorithms, leading to the revelation of crucial insights from data. This process facilitates the acquisition

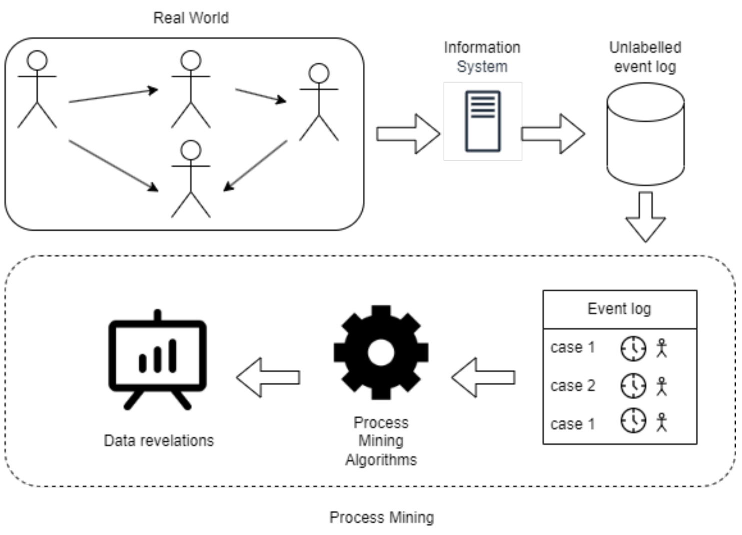


Figure 1: Process mining overview (adapted from [6]).

of key business process perspectives, including performance, data, organizational (resources), and control-flow.

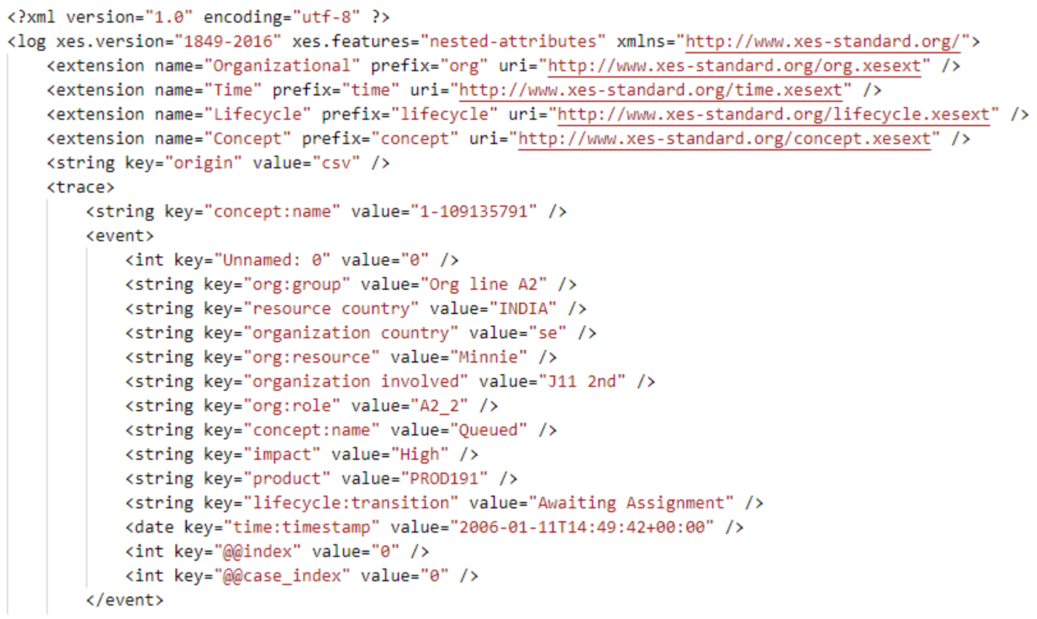
The yielded results extend their utility beyond their visualization. They offer insights, identifying bottlenecks and deviations, and even anticipating and diagnosing performance and compliance issues. This approach is adaptable across a broad spectrum of organizations and industries, underlining the wide applicability of PM principles.

2.1.2 *Event Log*

An event log is a record of events that describes all the activities (events) registered during the execution of a business process. In order to be effective, a log should contain certain key pieces of information:

- **Case ID**: A unique identifier assigned to each business process instance or case within the log;
- **Event**: Descriptive label for the event or activity being recorded;
- **Start Date**: Indicating the precise moment when the event commenced;
- **End Date**: Marking the conclusion of the event.

While event logs can include additional attributes to provide more context, these four attributes are considered essential for accurately and effectively tracking and analysing the events recorded in the log.



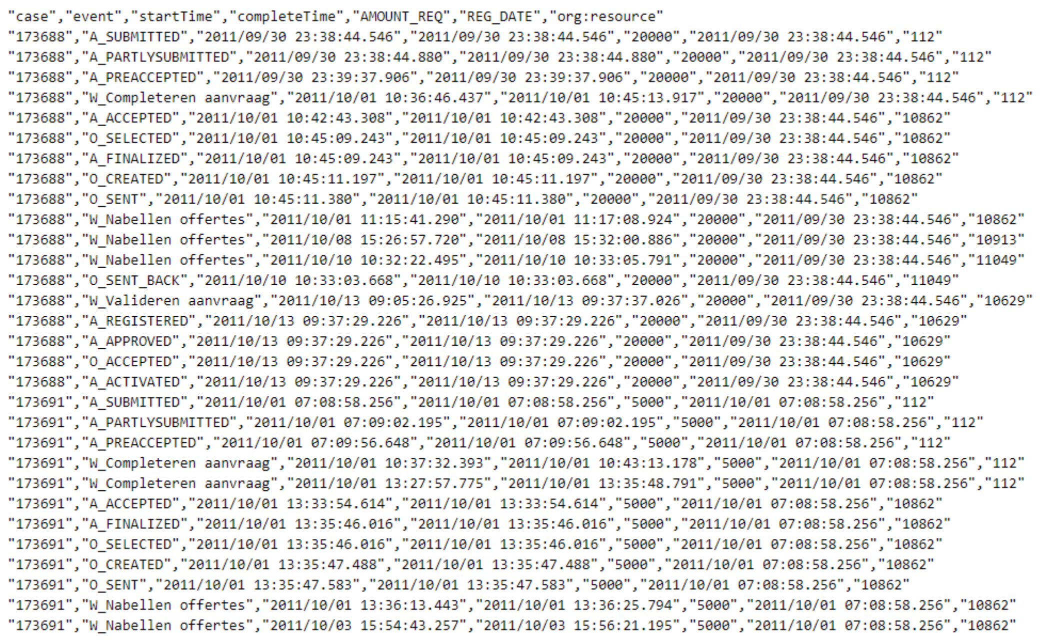
```
<?xml version="1.0" encoding="utf-8" ?>
<log xes.version="1849-2016" xes.features="nested-attributes" xmlns="http://www.xes-standard.org/">
    <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext" />
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext" />
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext" />
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext" />
    <string key="origin" value="csv" />
    <trace>
        <string key="concept:name" value="1-109135791" />
        <event>
            <int key="Unnamed: 0" value="0" />
            <string key="org:group" value="Org line A2" />
            <string key="resource country" value="INDIA" />
            <string key="organization country" value="se" />
            <string key="org:resource" value="Minnie" />
            <string key="organization involved" value="J11 2nd" />
            <string key="org:role" value="A2_2" />
            <string key="concept:name" value="Queued" />
            <string key="impact" value="High" />
            <string key="product" value="PROD191" />
            <string key="lifecycle:transition" value="Awaiting Assignment" />
            <date key="time:timestamp" value="2006-01-11T14:49:42+00:00" />
            <int key="@@index" value="0" />
            <int key="@@case_index" value="0" />
        </event>
```

Figure 2: Example of an XES file, displaying a partial view.

The standard format for an event log is the eXtensible Event Stream (XES)[1] format, which is supported by the majority of PM tools. This format is a standard that was adopted in 2010 by the IEEE Task Force on PM, and it became an official Institute of Electrical and Electronics Engineers (IEEE) standard in 2016 [7]. Figure 2 displays a XES file configuration, illustrating its key components for PM. The file encompasses crucial attributes relevant to process analysis, such as resource details, roles, timestamps, and product information, among others. This structured format captures the essential data needed for process exploration, analysis, and optimization.

In between the extraction of data from a data source and the creation of an event log in the XES, it is common to use Comma-Separated Values (CSV) as an intermediate format. CSV is a format used for storing tabular data, in which the first row normally represent the name of the field of each column and the following rows are for values. The values in the fields are separated by commas. CSV files are commonly used for storing data from spreadsheets and databases, and are often used for importing and exporting data between different software tools. Figure 3 presents a sample of a CSV file used in the project, specifically sourced from [8].

1 https://xes-standard.org/



```
"case","event","startTime","completeTime","AMOUNT_REQ","REG_DATE","org:resource"
"173688","A_SUBMITTED","2011/09/30 23:38:44.546","2011/09/30 23:38:44.546","20000","2011/09/30 23:38:44.546","112"
"173688","A_PARTLYSUBMITTED","2011/09/30 23:38:44.880","2011/09/30 23:38:44.880","20000","2011/09/30 23:38:44.546","112"
"173688","A_PREACCEPTED","2011/09/30 23:39:37.906","2011/09/30 23:39:37.906","20000","2011/09/30 23:38:44.546","112"
"173688","W_Completeren aanvraag","2011/10/01 10:36:46.437","2011/10/01 10:45:13.917","20000","2011/09/30 23:38:44.546","112"
"173688","A_ACCEPTED","2011/10/01 10:42:43.308","2011/10/01 10:42:43.308","20000","2011/09/30 23:38:44.546","10862"
"173688","O_SELECTED","2011/10/01 10:45:09.243","2011/10/01 10:45:09.243","20000","2011/09/30 23:38:44.546","10862"
"173688","A_FINALIZED","2011/10/01 10:45:09.243","2011/10/01 10:45:09.243","20000","2011/09/30 23:38:44.546","10862"
"173688","O_CREATED","2011/10/01 10:45:11.197","2011/10/01 10:45:11.197","20000","2011/09/30 23:38:44.546","10862"
"173688","O_SENT","2011/10/01 10:45:11.380","2011/10/01 10:45:11.380","20000","2011/09/30 23:38:44.546","10862"
"173688","W_Nabellen offertes","2011/10/01 11:15:41.290","2011/10/01 11:17:08.924","20000","2011/09/30 23:38:44.546","10862"
"173688","W_Nabellen offertes","2011/10/08 15:26:57.720","2011/10/08 15:32:00.886","20000","2011/09/30 23:38:44.546","10913"
"173688","W_Nabellen offertes","2011/10/10 10:32:22.495","2011/10/10 10:33:05.791","20000","2011/09/30 23:38:44.546","11049"
"173688","O_SENT_BACK","2011/10/10 10:33:03.668","2011/10/10 10:33:03.668","20000","2011/09/30 23:38:44.546","11049"
"173688","W_Valideren aanvraag","2011/10/13 09:05:26.925","2011/10/13 09:37:37.026","20000","2011/09/30 23:38:44.546","10629"
"173688","A_REGISTERED","2011/10/13 09:37:29.226","2011/10/13 09:37:29.226","20000","2011/09/30 23:38:44.546","10629"
"173688","A_APPROVED","2011/10/13 09:37:29.226","2011/10/13 09:37:29.226","20000","2011/09/30 23:38:44.546","10629"
"173688","O_ACCEPTED","2011/10/13 09:37:29.226","2011/10/13 09:37:29.226","20000","2011/09/30 23:38:44.546","10629"
"173688","A_ACTIVATED","2011/10/13 09:37:29.226","2011/10/13 09:37:29.226","20000","2011/09/30 23:38:44.546","10629"
"173691","A_SUBMITTED","2011/10/01 07:08:58.256","2011/10/01 07:08:58.256","5000","2011/10/01 07:08:58.256","112"
"173691","A_PARTLYSUBMITTED","2011/10/01 07:09:02.195","2011/10/01 07:09:02.195","5000","2011/10/01 07:08:58.256","112"
"173691","A_PREACCEPTED","2011/10/01 07:09:56.648","2011/10/01 07:09:56.648","5000","2011/10/01 07:08:58.256","112"
"173691","W_Completeren aanvraag","2011/10/01 10:37:32.393","2011/10/01 10:43:13.178","5000","2011/10/01 07:08:58.256","112"
"173691","W_Completeren aanvraag","2011/10/01 13:27:57.775","2011/10/01 13:35:48.791","5000","2011/10/01 07:08:58.256","112"
"173691","A_ACCEPTED","2011/10/01 13:33:54.614","2011/10/01 13:33:54.614","5000","2011/10/01 07:08:58.256","10862"
"173691","A_FINALIZED","2011/10/01 13:35:46.016","2011/10/01 13:35:46.016","5000","2011/10/01 07:08:58.256","10862"
"173691","O_SELECTED","2011/10/01 13:35:46.016","2011/10/01 13:35:46.016","5000","2011/10/01 07:08:58.256","10862"
"173691","O_CREATED","2011/10/01 13:35:47.488","2011/10/01 13:35:47.488","5000","2011/10/01 07:08:58.256","10862"
"173691","O_SENT","2011/10/01 13:35:47.583","2011/10/01 13:35:47.583","5000","2011/10/01 07:08:58.256","10862"
"173691","W_Nabellen offertes","2011/10/01 13:36:13.443","2011/10/01 13:36:25.794","5000","2011/10/01 07:08:58.256","10862"
"173691","W_Nabellen offertes","2011/10/03 15:54:43.257","2011/10/03 15:56:21.195","5000","2011/10/01 07:08:58.256","10862"
```

Figure 3: CSV example, displaying a partial view.

2.1.3 *Process Mining tools*

PM tools are becoming more and more prevalent as businesses look to improve their operations. These tools use data mining algorithms to analyse data from business processes, providing valuable insights that can help organizations streamline their operations and make more informed decisions. As the demand for PM tools grows, competition in the market is also increasing, with more and more companies adding PM features to their existing software or developing new tools to meet the needs of businesses.

In this work, we have explored the following PM tools: ProM[2], Disco[3] and PM4Py[4].

ProM is a general-purpose PM framework implemented in Java. It offers a wide range of tools and plug-ins for various PM tasks, including process discovery, process conformance checking, and process performance analysis. ProM has a large community of developers and users, and offers a range of support resources such as documentation and forums.

PM4Py is a PM library implemented in Python. It provides a range of PM algorithms and tools that can be used to discover, analyse, and improve business

2 https://www.promtools.org/
3 https://fluxicon.com/disco/
4 https://pm4py.fit.fraunhofer.de/

processes. PM4Py is designed to be developer-friendly and easy to integrate with other Python libraries and frameworks. It also includes a number of visualization tools for presenting PM results in a clear and intuitive way.

Disco is also a PM tool implemented in Java. It offers similar capabilities to ProM and PM4Py, including process discovery, process conformance checking, and process performance analysis. One of the key benefits of Disco is its ability to handle large amounts of data efficiently, making it well-suited for use with very large datasets. Disco also includes a number of visualization tools for presenting PM results in a clear and intuitive way.

Disco and ProM automatically suggest a column as the Case ID during the event log importation, primarily based on the column's name, especially when it includes "case" in its name. PM4Py does not offer an automatic Case ID identification feature. Additionally, a ProM plugin [9] has been developed for Case ID column identification during event log import.

2.2 PROBLEM STATEMENT

PM is a technique that is becoming increasingly popular as a way to improve business processes in a variety of industries. As more and more opportunities to use PM are discovered, people with all kinds of knowledge and expertise are starting to explore how they can leverage this approach to improve their processes and operations.

The current event log creation process involves a manual selection of the Case ID column. This selection requires users to have a deep understanding of the business processes, database structure, and domain-specific knowledge. This process can be challenging when dealing with event logs containing many columns.

Some people who are less familiar with PM may struggle to effectively use it due to their lack of knowledge and understanding of the technique. Despite this difficulty, they may still be interested in using PM to improve their processes, but may need to invest more time and effort into learning about the approach and how to apply it effectively.

Additionally, the goal of this project is to develop a library that, at the end, allows assisting users in making informed decisions regarding the selection of the column to serve as a process Case ID. The library aims to facilitate the generation of XES files, streamlining the process of preparing data for PM analysis.

Table 1: Paper characteristics.

Paper	Data Quality	Attribute Identification
[10]	X	
[11]	X	
[12]		X
[13]		X
[14]		X
[9]		X
[15]		X
[16]		X
[17]		X
[18]		X

2.3 RELATED WORK

This section details the process of identifying and reviewing relevant literature for the research about this project. The criteria for selecting these related works are also explained, along with a description of each individual study. A summary table is included, highlighting the coverage of the closest studies in relation to the theme of the project. This information helps to provide context and background for the research being presented, and demonstrates how the current study builds upon or extends previous work in the field.

In conducting the literature review for this research project, we used a variety of online tools and databases, including Google Scholar and Research Gate. We conducted keyword searches for terms such as “Process Mining”, “Attribute Identification”, “Data Quality” and “Log” in order to identify relevant articles and studies. We carefully reviewed the abstracts of these articles to assess their potential relevance and value for our research. Those studies that seemed particularly relevant or promising were saved for further consultation as needed. This process helped us to identify and select a range of relevant literature to inform our research.

Table 1 provides a summary of the key characteristics of the main studies that were investigated as part of the literature review. This summary table is intended to provide an overview of the relevant literature and to highlight the key contributions of each study.

2.3.1 *Data Quality*

In [11], the authors propose a process for improving data quality that involves understanding data quality requirements, measuring data quality, and promoting data quality awareness. This process is intended to help organizations ensuring that the data they use for PM is of high quality, which is essential for obtaining reliable insights and making informed decisions. The process starts by understanding the data quality requirements for the specific PM application, which helps to identify potential sources of data quality issues and prioritize efforts to address them. The process then involves measuring data quality using appropriate metrics and tools, which helps to identify and quantify any existing data quality issues.

In [10], the author examines various types of data quality taxonomies that classify the quality of data represented in the log, to then make use of an open source data quality assessment tool R-package DaQAPO to verify the quality of data. It also foresees the use of data cleaning heuristics for fixing incorrect timestamps, missing case identifiers, missing events and incorrect/missing attribute values.

2.3.2 *Attribute identification*

In [16]–[18], decision trees serve as the foundational method for Case ID identification within event logs. However, what sets them apart is their distinct approaches to this task, all hinging on the utilization of event column information. These studies adopt varying decision tree-based approaches to Case ID identification within event logs, using techniques such as filtering, cycling, and heuristics. Despite their use of decision trees, these papers emphasize the flexibility of this approach in case of ID identification, underlined by the role played by information from the event column.

In [12]–[14], authors investigate different approaches for Case ID identification within event logs. All stress the necessity of event-related information in this context. In [12], Neural Networks are applied with a focus on utilizing event data. Likewise, [13] employs an event pattern-based method, emphasizing the relevance of event-related data for Case ID identification. Lastly, in [14], non-overlaping sequence partitioning is explored as a means of identifying Case IDs. These studies collectively contribute to the variety of techniques available for Case ID identification, underscoring the essential nature of event information in this process.

Table 2: Papers of case id selection summary.

Paper	Identity		Method	Information
	Case ID	Event		
[19]	X	X	CNN	
[12]	X		Neural Networks	Event
[13]	X		Distinct event pattern	Event
[14]	X		Sequence partitioning	Event
[9]	X		ICI	Timestamp and Event
[15]	X		Case notion discovery	Timestamp and Event
[16]	X		Decision Trees	Event
[17]	X		Decision Trees	Event
[18]	X		Decision Trees	Event

In papers [9], [15], researchers investigate strategies for case ID identification within event logs, emphasizing the importance of both timestamp and event-related information. In [9], the authors employ ICI (Inter-Case Information) for case ID identification, while [15] explores Case Notion Discovery. These studies contribute to the field by highlighting the need for both timestamp and event information in their methodologies.

In [19], the authors presented a new approach for automatic attribute selection in event logs using a CNN. The CNN is able to classify four key PM entities: case ID, activity, originator label, and attribute label. This approach represents a significant advance in the field of PM, as it allows for the automatic identification of important attributes in event logs, which can be used to improve process analysis and optimization efforts.

The input data for this CNN-based approach is derived from event data and undergoes a series of transformations. Initially, the event data is converted from a table format to an array. Subsequently, this array is further processed, including a conversion into RGB format. Notably, a key innovation in this approach is the utilization of Event Density Embedding (EDE). EDE involves a projection from the transformed data into linear and after nonlinear functions.

Table 2 presents a summary of all the key points discussed in this section. It provides a clear overview of the main findings from the literature on the topic, highlighting the methods from the studies reviewed, and helps to provide a clear and concise overview of the current state of knowledge on the topic. The "Information" column indicates whether the methods in the papers require related information.

2.4 CONCLUSION

The purpose of this chapter has been to review the existing literature in the field and providing an overview of the current state of knowledge. Through reviewing studies, articles, and research papers, we gained a better understanding of the significant contributions and advancements made in the field. While some progress has been made, there is still ample opportunity for growth and improvement in certain areas, presenting opportunities for future research. This conclusion summarizes the key findings and trends we have identified in our review of the literature, and discuss their implications for our research and the field as a whole.

We analyse event logs to uncover information for identifying key attributes. This addresses limitations and offers a comprehensive solution for PM.

Finally, this report presents a challenge focused on creating a comprehensive approach to identify the Case ID, a key attribute used for PM. The primary focus is on improving process accessibility and user-friendliness, which would be particularly beneficial for newcomers in the field of PM, while also assisting others. This initiative seeks to elevate the quality of event logs and enable more efficient PM analysis, thereby advancing the field's progress and delivering improved solutions for PM.

3 METHODOLOGY

The methodology employed in this study comprises two key aspects: project phases and development methodology. These components provide a systematic and organized approach to conducting the research and achieving the defined objectives.

3.1 PROJECT PHASES

To make the demonstration of the project's various phases more intuitive and straightforward, a diagram (Figure 4) was created, illustrating the workflow throughout the project's development. The phases are the following:

- **Datasets Collection** - Systematically gathering relevant event logs for the project, ensuring the inclusion of high-quality and representative data for analysis and modelling;
- **Data Understanding** - Exploring the available dataset and identifying the columns or variables that could potentially serve as the Case ID. Understanding the nature of the data, its structure, and any existing metadata that can aid in the selection process;
- **Data Preparation** - Cleansing and preprocessing the dataset to ensure data quality and consistency. Handling missing values, outliers, and any data anomalies that may impact the selection of the Case ID column;
- **Metric Selection** - Identifying and prioritizing relevant metrics aligned with the project objectives, considering factors such as measurability, interpretability, and potential impact. The selected metrics provide valuable insights for effective evaluation and decision-making;
- **Modelling** - Developing and implementing attribute selection methods that can effectively identify the Case ID column from the dataset. This involves applying various techniques and algorithms to determine the most suitable attribute(s) for the purpose of PM;

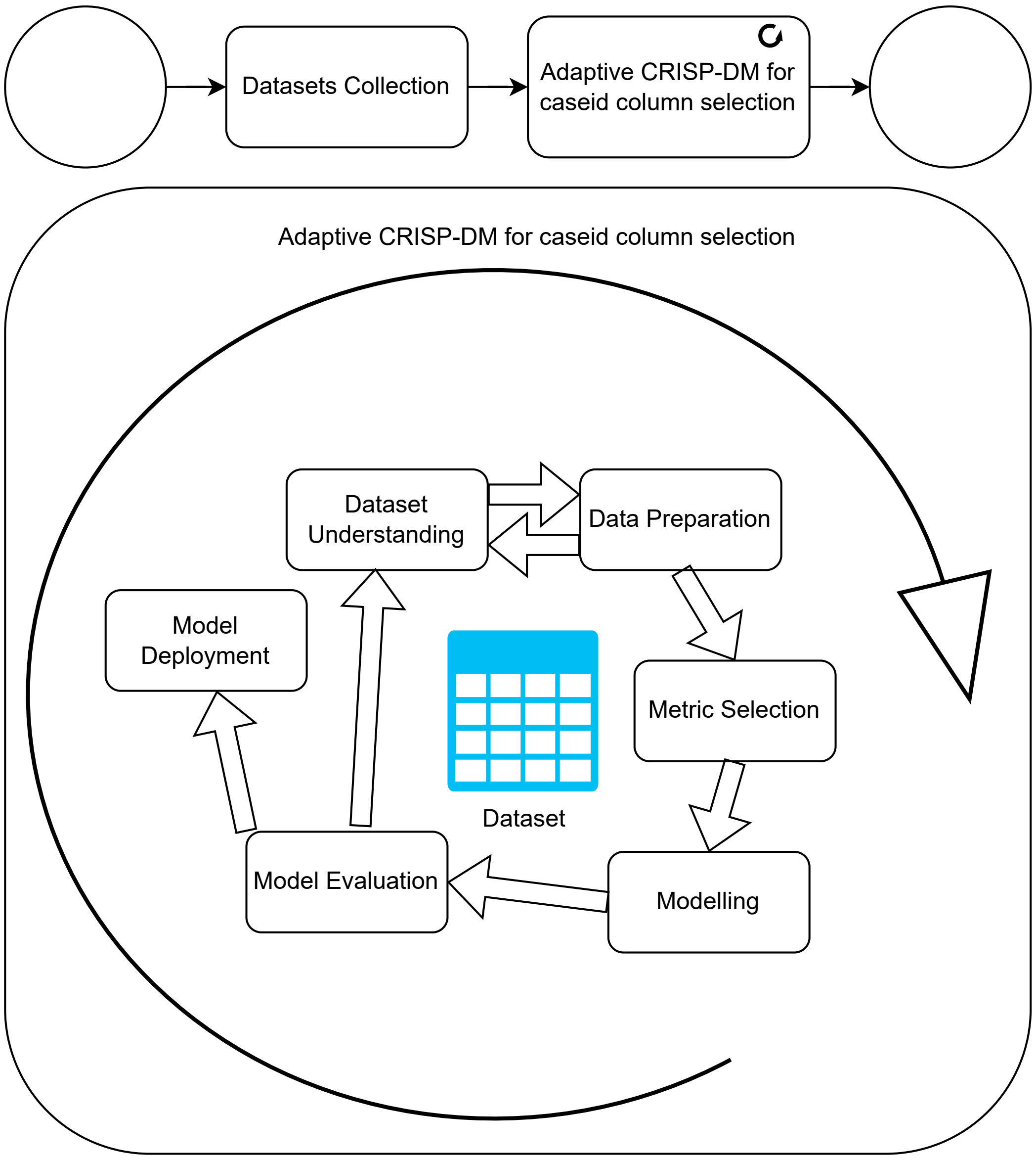


Figure 4: Project phases.

- **Model Evaluation** - Evaluating the candidate model based on predefined criteria. Considering the uniqueness and stability of the values in each column, as well as the model's ability to uniquely identify each case or record in the dataset;
- **Model Deployment** - After establishing a model for identifying the Case ID column, implementing it within data analysis and PM tools for effective use.

3.2 DEVELOPMENT METHODOLOGY

Our methodology revolves around a structured weekly meeting format to track progress and address challenges promptly. These meetings involve active participation from the research team and advisors to facilitate knowledge sharing and issue resolution.

During these meetings, we conduct a comprehensive review of the previous week's progress, identifying any encountered obstacles and collectively devising solutions. The discussions delve deep into research findings and emerging trends, fostering critical analysis and stimulating innovative thinking. This iterative process culminates in the establishment of well-defined objectives for the upcoming week.

This methodology places a strong emphasis on flexibility, responsiveness, and adaptability. It allows us to seamlessly integrate new insights, fine-tune our research methods, and explore additional research avenues, all of which significantly contribute to the overall success of the project.

3.3 DEVELOPMENT ENVIRONMENTS

In order to support the development and execution of our research project, we established specific development environments that provided the necessary hardware resources and software configurations. These environments were designed to accommodate the computational requirements of our analysis and ensure the reproducibility of our experiments.

The key hardware components of our development environments are included in Table 3.

Table 3: PC specs.

CPU	Intel i7-10750H
RAM	16GB 2667 MHz
GPU	NVIDIA GeForce RTX 2060

Listing 1: Jupiter CPU Service Configuration.

```
version: "3.8"
jupiter:
        shm_size: '5gb'
        security_opt:
         - seccomp:unconfined
        deploy:
                resources:
                        limits:
                                cpus: '5.0'
                                memory: 7GB
                        reservations:
                                cpus: '5'
                                memory: 4GB
```

This hardware configuration provided substantial computational power, enabling us to process and analyse large datasets, perform complex machine learning tasks, and leverage GPU acceleration for certain algorithms.

To ensure reproducibility and ease of deployment, we employed Docker and Docker Compose to create consistent development environments across different systems. Docker containers encapsulated the necessary software dependencies, libraries, and configurations, allowing for seamless replication and distribution of our research environment.

We present two example Docker Compose setups that illustrate the composition of our development environments in Listing 1 and Listing 2.

In Listing 1, we used an image from jupyter/datascience-notebook to provide a Jupyter environment with Central Processing Unit (CPU) support. The container was configured with appropriate volumes for library scripts, datasets, glscpu notebooks, and result storage. The service had access to the specified hardware resources, including 5 CPU cores and 11GB of memory.

For Listing 2, we used an image from rapidsai/rapidsai:cuda11.8-runtime-ubuntu22 .04-py3.10, which included Graphics Processing Unit (GPU) support for accelerated computations. The container had access to volumes for library scripts, datasets, GPU notebooks, result storage, cache, logs, and Artificial Intelligence (AI). The setup was configured to use 5 CPU cores, 11GB of memory, and an NVIDIA GeForce RTX 2060 GPU.

Listing 2: Jupiter CPU Service Configuration.

```
version: "3.8"
jupitergpu:
        shm_size: '5gb'
        security_opt:
         - seccomp:unconfined
        deploy:
                resources:
                        limits:
                                cpus: '5.0'
                                memory: 7GB
                        reservations:
                                cpus: '5'
                                memory: 4GB
                                devices:
                                        - driver: nvidia
                                                      count: 1
                                                capabilities: [gpu]
```

4 DATA COLLECTION, UNDERSTANDING AND PREPARATION

The data understanding process is a fundamental stage in any data analysis or research project. It involves selecting the appropriate event logs, exploring its characteristics, and preparing it for further analysis. This chapter serves as a comprehensive overview of the data understanding process undertaken in this study. We selected a dataset that aligns with our research objectives and provides the necessary information to answer our research questions. Through exploratory data analysis, we gained valuable insights into the dataset, including its size, structure, and temporal context. We assessed the data quality, identified patterns and trends, and explored potential relationships among variables. The insights and preparations made in this chapter lay the foundation for the subsequent chapters, where we apply specific methodologies and conduct in-depth analyses to extract valuable insights and findings from the data.

4.1 SELECTED EVENT LOGS DATASET

The selected event logs dataset for this study plays a crucial role in addressing the research objectives and is derived from various reputable sources. In the state-of-the-art literature review, it was observed that the majority of the referenced papers use event logs from the Business Process Intelligence (BPI) Challenge. Therefore, to ensure the comprehensiveness of our analysis, we collected all available BPI Challenge event logs, except for those from 2014 and 2016, which were excluded due to being in the German language and for the process mining attributes being unlabelled.

Additionally, to implement and evaluate our proposed solution, we incorporated an event log from an ongoing project related to the manufacturing of moulds. This event log is of significant importance as it reflects the real-world context of the project and enables us to validate the effectiveness of our approach in a practical setting.

By combining the BPI Challenge event logs and the proprietary event log, we aim to leverage a diverse range of business process data and foster a comprehensive understanding of process behaviour, patterns, and performance. The inclusion of these event logs facilitates robust analysis and enhances the applicability and generalizability of our findings to real-world scenarios. Table 5 provides a summary of the event log names and a brief description of what they represent.

Table 4: Event Logs Description.

Name	Short Description	Reference
BPI 2011	Real life log of a Dutch academic hospital	[20]
BPI 2012	Event log of a loan application process	[8]
BPI 2013 closed problems	Logs of Volvo IT incident and problem management	[21]
BPI 2013 incidents		[22]
BPI 2013 open problems		[23]
BPI 2015 1	Logs of five Dutch municipalities	[24]
BPI 2015 2		[25]
BPI 2015 3		[26]
BPI 2015 4		[27]
BPI 2015 5		[28]
BPI 2017	Event log pertains to a loan application process of a Dutch financial institute	[29]
BPI 2018	Event log of the handling of applications for EU direct payments	[30]
BPI 2019	Event log from a large multinational company operating from The Netherlands in the area of coatings and paints	[31]
BPI 2020 domestic declarations		[32]
BPI 2020 international declarations		[33]

Name	Short Description	Reference
BPI 2020 pre-paid travel costs	Logs from a University for request, permits, and cost	[34]
BPI 2020 request for payment		[35]
BPI 2020 travel permits		[36]
Moulds	Log from moulds production	

4.2 DATA UNDERSTANDING

In this section, we begin by examining the complexity of the datasets used in our analyses. The complexity is assessed based on factors such as the number of rows, number of columns, event log time span, and file size. Table 5 provides a summary of these characteristics for each event log.

Analysing the number of rows in each event log gives us an indication of the amount of data available for analysis. For instance, the BPI 2018 event log contains an extensive 2,514,266 rows, suggesting a substantial volume of instances to be explored. On the other hand, the Moulds event log consists of 32,512 rows, indicating a relatively smaller sample size.

The number of columns in an event log provides insights into the dimensionality and complexity of the data. For example, the BPI 2020 travel permits event log contains a considerable 174 columns, suggesting a rich set of variables that may contribute to the complexity of the event log. Conversely, the BPI 2012 event log consists of only six columns, indicating a more streamlined set of attributes.

Furthermore, the event log time span represents the duration covered by the data, offering insights into the temporal context of the analysis. The BPI 2011 event log spans 1,172 days, indicating data collected over a considerable period. In contrast, the BPI 2017 event log covers a shorter time span of 397 days, implying a more focused temporal scope.

Considering the size of the BPI 2018 event log, which is approximately 1.75 GB, it becomes evident that analysing such a large event log may require us to adapt our approach to ensure efficient processing within a reasonable timeframe.

Table 5: Summary of Event logs.

Event Log Name	**Number of Rows**	**Number of Columns**	**Event Log Time Span**	**Size**
BPI 2011	150,291	128	1,172 days	79.2 MB
BPI 2012	150,291	6	165 days	18.9 MB
BPI 2013 closed problems	6,660	13	2,332 days	758 KB
BPI 2013 incidents	65,533	13	783 days	7.23 MB
BPI 2013 open problems	2,351	12	2,047 days	250 KB
BPI 2015 1	52,217	29	1,761 days	14.2 MB
BPI 2015 2	44,354	28	1,709 days	16.5 MB
BPI 2015 3	59,681	29	1,889 days	16.0 MB
BPI 2015 4	47,293	29	1,933 days	18.2 MB
BPI 2015 5	59,083	29	1,926 days	23.7 MB
BPI 2017	561,671	19	397 days	129 MB
BPI 2018	2,514,266	75	1,356 days	1.75 GB
BPI 2019	1,595,923	22	26,372 days	519 MB
BPI 2020 domestic	56,437	11	889 days	12.6 MB
BPI 2020 international	72,151	24	1,313 days	33.6 MB
BPI 2020 prepaid travel	18,246	23	772 days	8.03 MB
BPI 2020 request for payment	36,796	15	941 days	11.3 MB
BPI 2020 travel permits	86,581	174	1,792 days	51.2 MB
Moulds	32,512	24	343 days	6.41 MB

To address this challenge, we transitioned from a CPU-based Jupyter environment to a GPU-accelerated environment, which leverages the power of the GPU and the cuDF[1] library. This transition enables us to process the dataset more efficiently and significantly reduce the processing time.

To provide a comparison of the impact of CPU and GPU processing on tasks involving data calculations, we conducted analysis on the BPI 2018 event log. Using the CPU-based Jupyter environment, the analysis took 8 minutes. These initial processing times using CPU alone served as benchmarks for our subsequent analysis.

After transitioning to the GPU-accelerated environment, we re-conducted the analysis on the event log. The processing time was reduced to 46 seconds, showcasing the significant time savings achieved through the utilization of the GPU and cuDF.

1 https://docs.rapids.ai/api/cudf/stable/

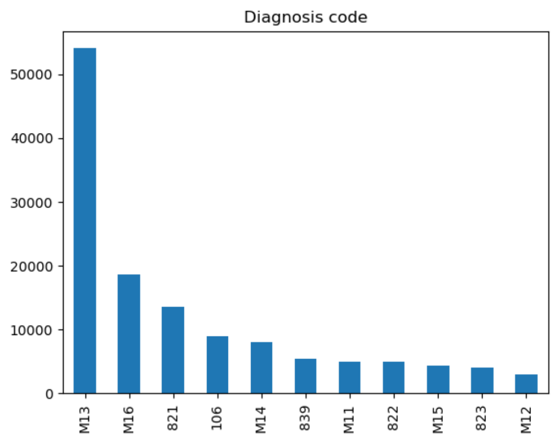


Figure 5: Frequency graph of the Diagnosis code attribute in the BPI 2011 event log. The x-axis represents the attribute values, while the y-axis indicates the frequency count.

By leveraging the computational power of the GPU and employing the cuDF library, we were able to expedite the processing of both small and large event logs. This accelerated processing time allows us to perform more comprehensive analysis and iterate more rapidly on our research objectives.

Understanding the complexity of the dataset is essential for framing our subsequent analysis appropriately. By considering these factors, we can make informed decisions regarding the choice of analytical techniques, potential challenges, and the significance of the findings.

For a thorough understanding of the dataset, we conducted exploratory data analysis techniques to gain insights into the data as a whole in each event log. The following analyses were performed:

- **Frequency Charts**: We created frequency charts for each attribute across the event logs (see Figure 5). These charts allowed us to visualize the occurrence and distribution of different attribute values, enabling us to identify any major trends or imbalances.
- **Box Plots**: Box plots were generated to detect the presence of outliers across the event logs (see Figure 6). By analysing the distribution of data points and the position of outliers, we gained insights into the potential presence of extreme values that could impact our analysis.
- **Dispersion Charts**: Dispersion charts were utilized to explore the variability and spread of values within the datasets (see Figure 7 and 8). These charts

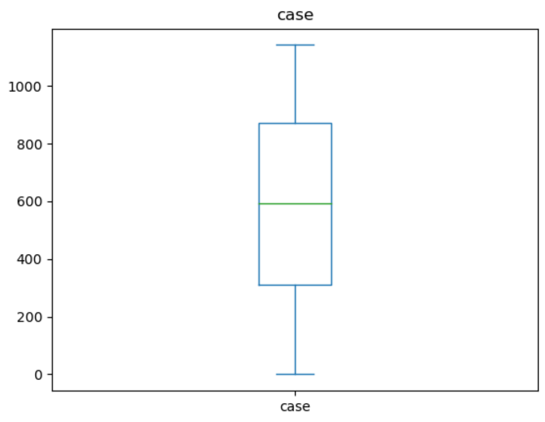


Figure 6: Box plot of the case attribute in the BPI Challenge 2011 event log, illustrating a balanced distribution with no outliers. The values range from 1 to 1000, and the quartiles are equally spaced.

enabled us to identify any patterns, clusters, or discrepancies in the data distribution.

- **Column Analysis**: We examined the number of columns in the event logs to understand their complexity and dimensionality. This analysis provided insights into the overall structure of the event logs and the number of variables available for further exploration. Figure 9 provides a comprehensive overview of our column analysis, missing values assessment, and unique values calculation.
- **Missing Values**: To assess the data quality, we determined the number of null values in each event log. This step helped us identify any missing data that would require imputation or handling during the preprocessing phase.
- **Unique Values**: We calculated the number of unique values for each event log. This analysis enabled us to identify categorical variables and assess the cardinality of different features.

By performing these exploratory analyses collectively, we gained a comprehensive understanding of the dataset, allowing us to proceed with further data processing and analysis in subsequent chapters.

Upon conducting exploratory analysis on the dataset using various techniques, we found that extracting the "Start Date" and "End Date" columns from the event logs was straightforward. This observation is significant because it does not require detailed knowledge about the event logs construction. The "Start Date" and "End Date" columns, capturing event initiation and completion dates, are vital for understanding temporal patterns in the data.

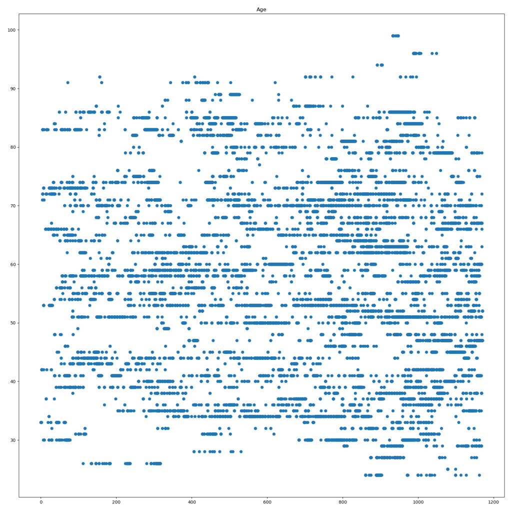


Figure 7: Dispersion Graph of BPI Challenge 2011 attribute age. The y-axis represents the age value, while the x-axis denotes the Start Time with an adjustment that subtracts the oldest recorded Start Time.

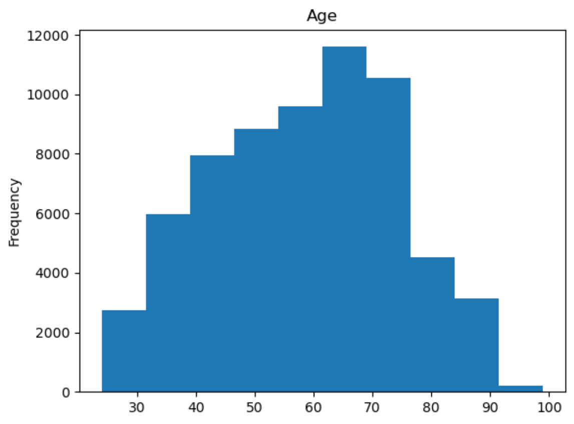
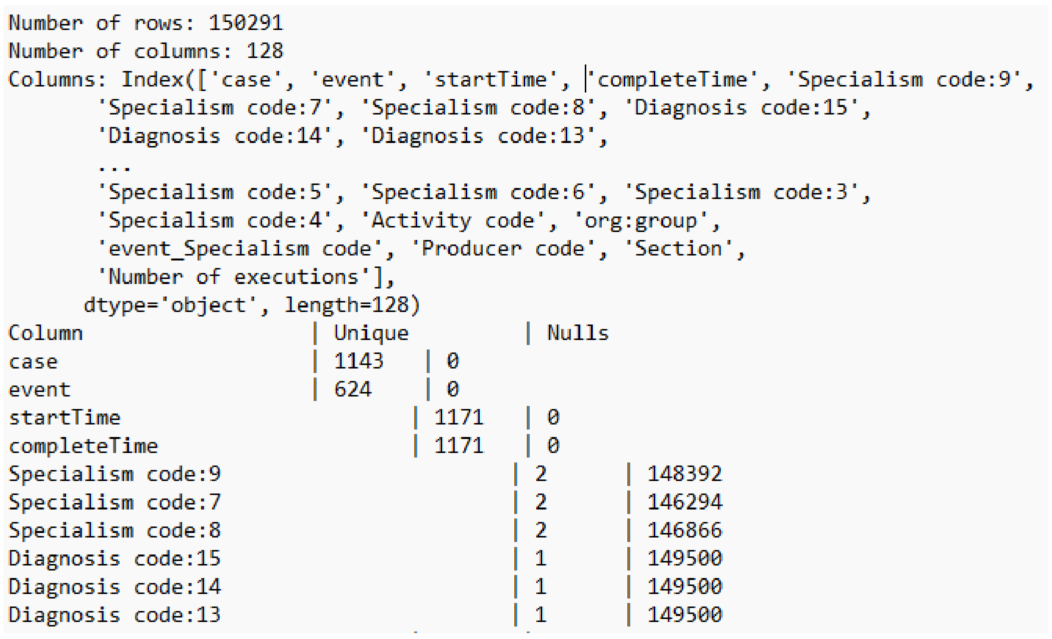


Figure 8: Histogram graph depicting the distribution of ages within the BPI Challenge 2011 event log. This graph illustrates the frequency of different age groups or values present in the event log, segmented into 10 distinct groups. The age range spans from 25 to almost 100, and the graph reveals any predominant age groups, outliers, or noteworthy trends in the data.



```
Number of rows: 150291
Number of columns: 128
Columns: Index(['case', 'event', 'startTime', 'completeTime', 'Specialism code:9',
       'Specialism code:7', 'Specialism code:8', 'Diagnosis code:15',
       'Diagnosis code:14', 'Diagnosis code:13',
       ...
       'Specialism code:5', 'Specialism code:6', 'Specialism code:3',
       'Specialism code:4', 'Activity code', 'org:group',
       'event_Specialism code', 'Producer code', 'Section',
       'Number of executions'],
      dtype='object', length=128)
Column                    | Unique         | Nulls
case                      | 1143    | 0
event                     | 624     | 0
startTime                         | 1171    | 0
completeTime                      | 1171    | 0
Specialism code:9                         | 2       | 148392
Specialism code:7                         | 2       | 146294
Specialism code:8                         | 2       | 146866
Diagnosis code:15                         | 1       | 149500
Diagnosis code:14                         | 1       | 149500
Diagnosis code:13                         | 1       | 149500
```

Figure 9: Representation of event log columns with nulls and uniques and basic information.

For the dates extraction process, we begin by obtaining an event log. Then, we launch a parsing procedure to identify columns containing date information in this event log. This parsing process methodically checks for various date formats, including timestamps, and converts them into timestamps for consistency. Afterward, we examine the data columns to locate a row in the event log where all date columns are fully populated. From this specific row, we extract the oldest date, known as the "Start Date", and the newest date, referred to as the "End Date". This systematic approach ensures the accurate retrieval of temporal information from the event log.

4.3 REFINED EXPLORATORY DATA ANALYSIS: START DATE ANALYSIS

In this section, we explore the impact of the "Start Date" column on the event logs and how it can serve as a pivotal point for further analysis. By considering the temporal dimension of the data, we aim to uncover patterns, trends, and correlations that might have been overlooked in the previous analysis. This refined approach allows us to delve deeper into the event logs and extract meaningful information related to time-based dynamics and relationships.

We begin by revisiting the event logs and examining the "Start Date" column in detail. We analyse the distribution of start dates, identify any anomalies or discrepancies, and gain a comprehensive understanding of the temporal context of the data. With this foundation, we proceed to explore the interplay between the

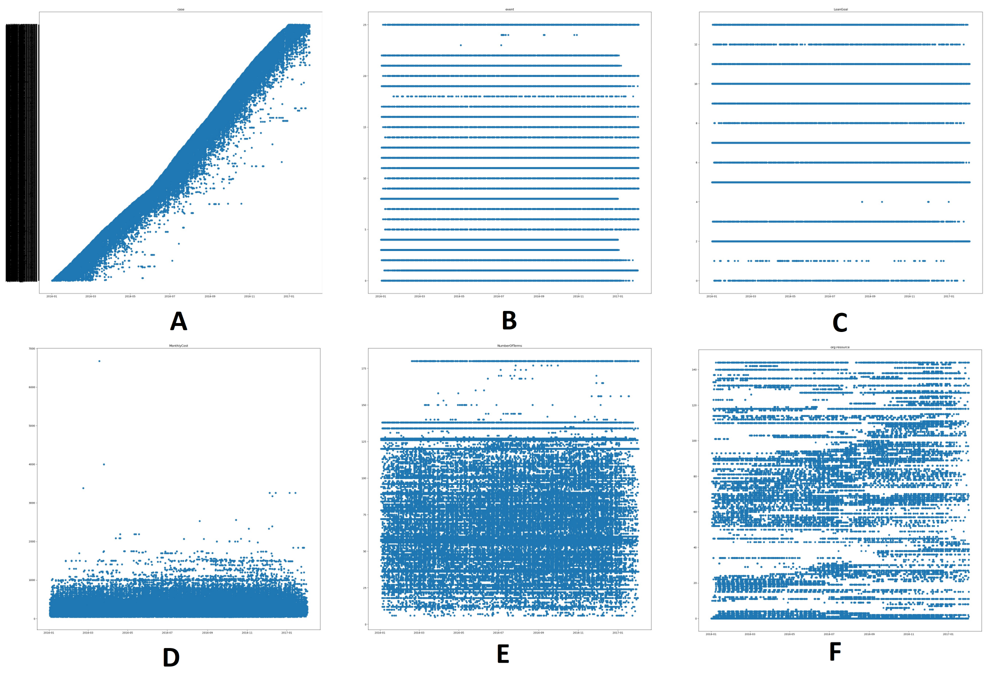


Figure 10: Example graphics with start date as X-axis.

"Start Date" column and other attributes. We investigate how different variables interact with the start dates, potentially shedding light on causal relationships or dependencies.

Through this refined exploratory data analysis, we uncover new insights and enhance our understanding of the dataset. By leveraging the "Start Date" column as a pivot, we anticipate identifying temporal patterns, seasonality effects, and potential correlations that informed our subsequent analysis and contributed to the overall findings of our research.

To gain insights into the relationships between the "Start Date" column and other attributes, we created plot charts for each column of the event log, with the "Start Date" as the X-axis and the respective attribute values as the Y-axis. This visual representation allowed us to observe the distribution and behaviour of different variables over time. As we analysed the charts, a pattern emerged, suggesting the presence of a distinct temporal pattern across multiple attributes, as shown in Figure 10.

In Figure 10 there are six images labelled: A, B, C, D, E, and F. Image A represents the Case ID, while Image B represents the activity. The remaining images provide insights into various columns within a event log. These visualizations are derived from the [20] event log. Importantly, these graphics are generated based on

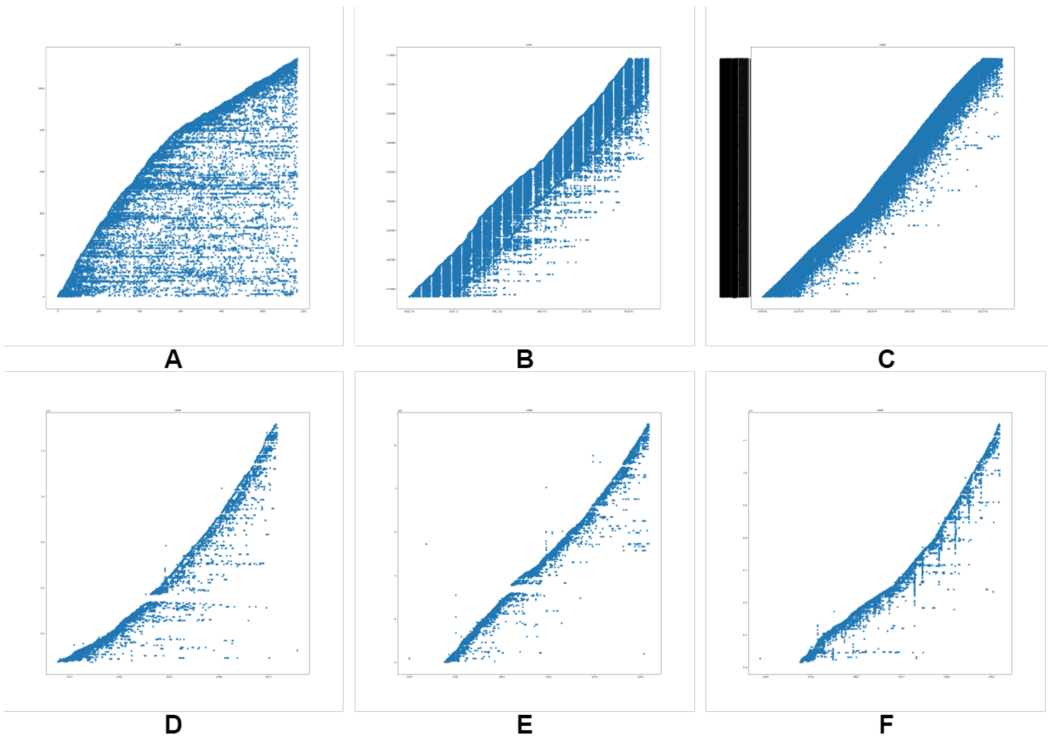


Figure 11: Example graphics of Case ID in different event logs.

the ordering of the “Start Date” column alongside the respective attribute columns. The Case ID image exhibits a different pattern than the other columns.

Building on this, we have generated Figure 11, comprising six labelled images: A, B, C, D, E, and F. These images represent the Case ID for event logs [20], [8], [29], [24], [26], [27]. These graphics were created using the same chronological ordering as the previous figure but without axes, facilitating pattern analysis.

In Figure 11, a noticeable recurring pattern emerges, characterized by a diagonal orientation at the starting point of each row. Particularly in images A and E, there is a distinct shape reminiscent of a triangle. As we understand, a process case inherently comprises a beginning and an end, and almost all images seem to illustrate that fundamental concept. This observation leads us to formulate the hypothesis that mature event logs, those containing numerous completed processes, tend to exhibit this diagonal pattern and “younger” event logs tend to exhibit a triangle pattern because they have numerous unfinished processes.

It is crucial to realize that the process timeline’s characteristics may be influenced by the duration of processes within the company and the specific type of company. These factors can contribute to variations in the observed patterns, highlighting the complex interplay between data characteristics and real-world processes.

Our data exploration led to an observation. We hypothesized that the median time span of values in the "Case ID" column is shorter compared to other columns in the event log. This insight contributed to the development of an expression outlined in Chapter 6. This expression forms the basis for the heuristic we used to identify the "Case ID" column. It also led us to the approach based on CNNs, where the model's input images are charts. The goal in this approach was to develop neural networks able to identify the pattern described above.

4.4 DATA PREPARATION

In this section, we outline the steps taken to prepare the dataset for further analysis. The data preparation process comprises two main steps, with each main step consisting of multiple sub-steps. The first main step is dedicated to the removal of specific columns:

1. Date Extraction: We begin by employing a date extraction procedure to identify the "Start Date" and "End Date" columns within the event log. This step involves parsing the data to locate date information, converting it into timestamps, and selecting the earliest date as the "Start Date" while discarding all other date columns;
2. Removing Empty Columns: To streamline the event log, we identify and remove columns that are entirely empty. These empty columns do not contribute to the analysis and can be safely eliminated;
3. Handling Columns with Two Unique Values: In some cases, columns with only two unique values may not provide significant information for our analysis. Therefore, we identify such columns and consider whether they should be retained or excluded from the event log;
4. Correlation-Based Column Aggregation: In this step, we use the Correlation-Based Column Aggregation method to identify columns with a correlation coefficient of 1. These columns are combined into single columns, simplifying the event log while retaining crucial information.

We perform these data preparation tasks using the Pandas[2] library, a data manipulation tool in Python that helps with data cleansing and transformation tasks.

2 https://pandas.pydata.org/

For the second step, we perform column preparation by creating duplicates of the "column" and "Start Date" columns. This serves as the final step, establishing the starting point for testing our hypothesis and generating images for the CNN:

1. Data Filtering: Initially, we filter the event log to include only the "Start Date" column and one additional specified column;
2. Null Value Exclusion: In the subsequent stage, we remove rows containing null values in either of these two columns, ensuring complete records remain for further analysis;
3. Data Sorting: To enhance data organization, we arrange the event log in a specific order. Firstly, we sort it chronologically based on the "Start Date" column, capturing the temporal progression of events. Then, we arrange the data based on the designated column, improving its structure for efficient attribute-based analysis;
4. Categorical Transformation: Finally, we optimize the event log for subsequent analysis by transforming the values within the designated column into a categorical type, providing a structured foundation for our research.

By implementing these data preparation steps, we ensure that the dataset is cleansed and optimized for subsequent analysis, enabling us to focus on the most relevant attributes and data for our research.

5 CONVOLUTIONAL NEURAL NETWORK MODEL

In this chapter, we explore the application of Convolutional Neural Networks (CNN) for Case ID column identification. CNNs are a class of deep learning models widely used in computer vision tasks, and we use them to automatically learn relevant patterns and features from our dataset.

5.1 UNDERSTANDING NEURAL NETWORKS AND CNNS

In this section, we delve into the world of neural networks, a core component of modern data science and deep learning algorithms. Neural networks, also known as Artificial Neural Networks (ANNs), are a subset of machine learning models used in deep learning. They are inspired by the human brain, mimicking the way biological neurons signal to each other [37].

Neural networks are typically organized as a series of layers of interconnected nodes, comprising an input layer, one or more hidden layers, and an output layer. Each node receives signals from nodes in the preceding layer. These signals undergo a weighted sum calculation, which is then passed through an activation function to produce the node's output. This architecture is visually represented in Figure 12. This process can be mathematically expressed as:

$$\text{Node Output} = \text{Activation}\left(\sum_i \text{Input}_i \times \text{Weight}_i\right)$$

Here, weights play a crucial role as they determine the strength of connections between nodes. During training, these weights are iteratively adjusted to minimize the difference between the network's predictions and the actual data, a process known as backpropagation. This training process continues until the network achieves the desired level of performance.

CNNs, a specialized subset of neural networks, are particularly effective for image-related tasks. They excel in automatically identifying patterns and features within

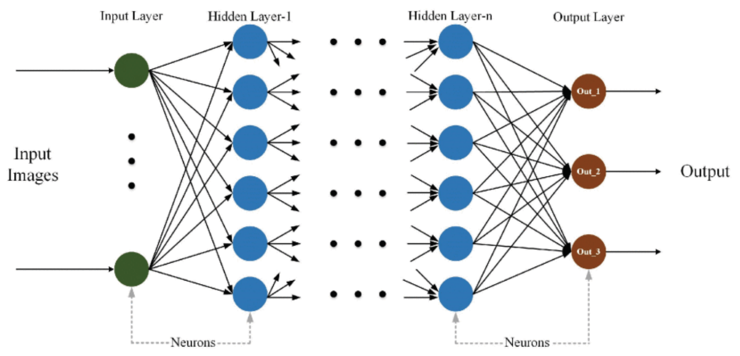


Figure 12: Illustration of the neural network architecture [38].

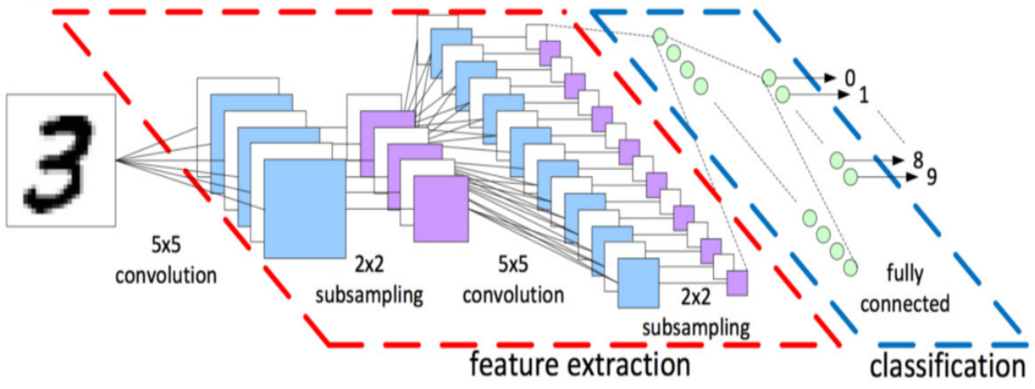


Figure 13: CNN Architecture example (adapted from [39]).

image data, revolutionizing fields like computer vision, image recognition, object detection, and autonomous driving.

A CNN consists of two main sections: feature extraction and classification. The feature extraction section identifies significant image features, like edges and textures, through convolution and pooling layers. The classification section utilizes these features for tasks like object identification or classification. Figure 13 provides an example of this architecture.

In order to allow training models with small datasets, it is common to use transfer learning. This technique accelerates learning for new tasks by utilizing pre-trained models that were trained on large datasets. It begins with a model that has already learned from a prior task. Transfer learning has two main methods: feature extraction and fine-tuning. These methods optimize the machine learning model creation process.

Feature extraction focuses on using the existing layers of the pre-trained model, especially the lower layers that identify basic patterns. These layers automatically extract essential features from the data, which become inputs for the task-specific model. On the other hand, fine-tuning involves adjusting the pre-trained model's layers to better fit the specific task. This typically includes unfreezing some or all of the layers and retraining them with the problem specific data. Fine-tuning refines the model's understanding of the task, especially when it differs significantly from the model's original training.

5.2 DATASET CREATION, PREPARATION AND PREPROCESSING

The images for training our CNN were generated from the event logs presented in Chapter 4. Each event log underwent data preparation to create corresponding scatter chart image samples for each column, where the X-axis corresponds to the "Start Date" column values and the Y-axis correspond to the values of the column we want to classify as case or not case, that is, as being a Case ID column or not. The images were generated using the Matplotlib pyplot library and rendered without the axes with an aspect ratio of 2500x2500 pixels. As an example, we have Figure 11, consisting of two groups of images, *case* and *not case*.

After generating the images, they were separated into two categories: "Case ID" and "Not Case ID". The "Case ID" category contains images corresponding to the "Case ID" column, while the "Not Case ID" category includes images corresponding to the other columns.

The dataset used for training and evaluating the CNN models is unbalanced, consisting of 19 "Case ID" images and 332 "Not Case ID" images. It is important to note that these 19 event logs each contain only one column/image corresponding to the Case ID column, hence the existence of only 19 images for this class. The remaining 332 images encompass all the columns from the event logs that are not the Case ID. To address the class imbalance, we employed a 4-fold cross-validation strategy without the creation of the test set.

For dataset creation, we randomly divided the images into four equal parts for the 4-fold approach, while maintaining their original case/not case ratio. In each fold, a balanced distribution of "Case ID" and "Not Case ID" images was ensured.

We decided not to use a separate test set, as cross-validation allows for a comprehensive evaluation of the model's performance while making efficient use of the available data. By using cross-validation, we can assess the model's generalization across different folds, providing more robust results and mitigating potential overfitting issues.

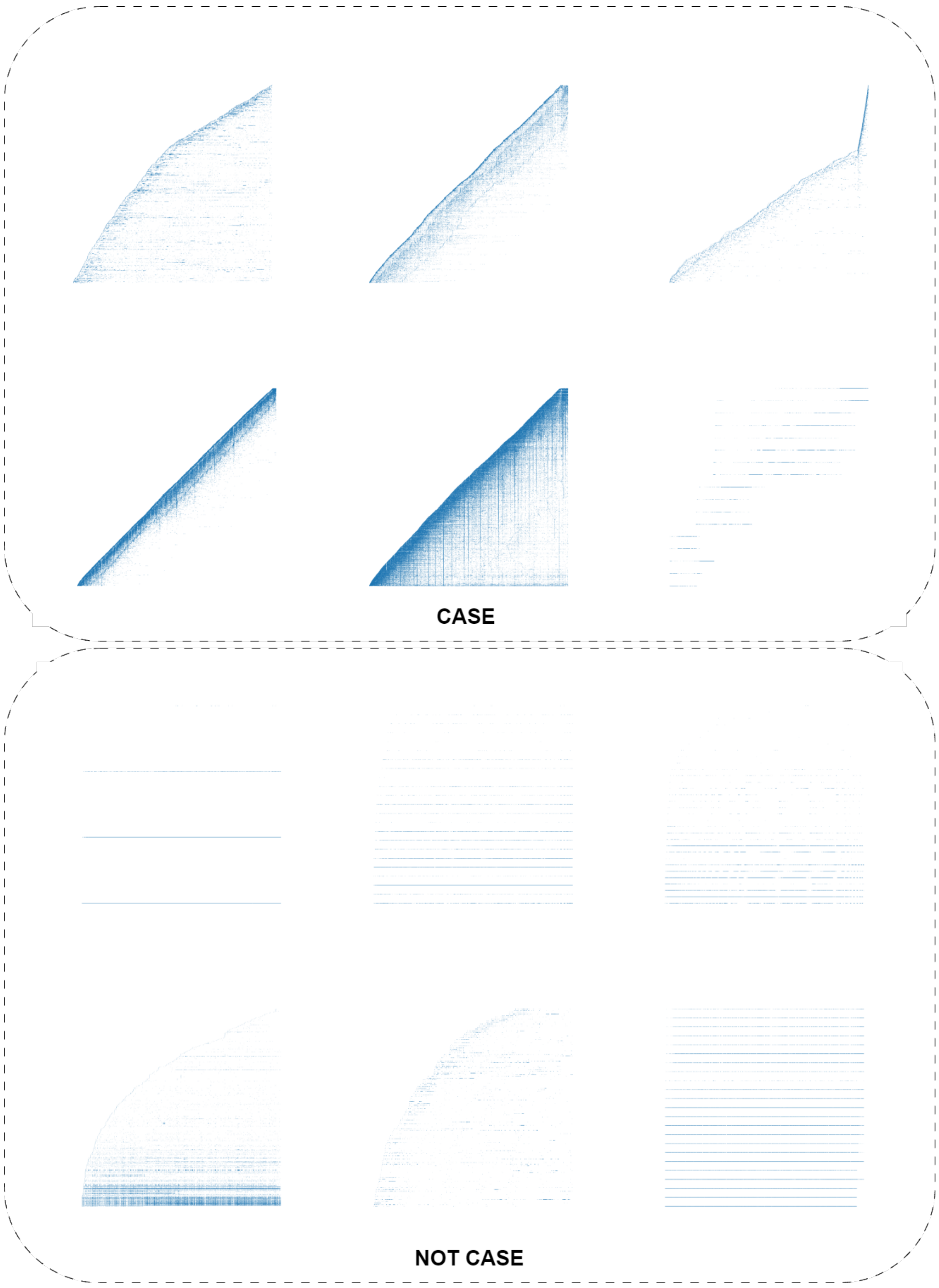


Figure 14: Example graphics of case and not case in different datasets.

To prepare the data for binary classification, the “Case ID” class was automatically assigned value 1, and the “Not Case ID” class was assigned value 0. This binary representation allows the model to distinguish between the two categories during training.

By performing data preprocessing, we ensure that the image data is appropriately formatted and ready for training the CNN models. The use of cross-validation strategies and the balanced distribution of “Case ID” and “Not Case ID” images during dataset creation contribute to a more comprehensive and accurate evaluation of the CNN models in identifying the Case ID column.

5.3 EVALUATION METRICS

When working with an unbalanced dataset, it is essential to use evaluation metrics that can effectively assess the performance of the model in such scenarios. Here, we discuss the evaluation metrics we employed to measure the performance of the generated CNN models in identifying the Case ID column:

Precision: Precision measures the proportion of true positive predictions among all positive predictions (i.e., cases identified correctly among all instances classified as cases). It is especially useful when the cost of false positives is high. Precision ranges from 0 to 1, where 1 represents perfect precision.

Recall (Sensitivity or True Positive Rate): Recall computes the proportion of true positive predictions among all actual positive instances (i.e., case ID columns correctly identified among all actual case ID columns). It is particularly important when the cost of false negatives is high, as it indicates the model’s ability to find all relevant instances. Recall ranges from 0 to 1, where 1 represents perfect recall.

F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of the model’s accuracy in classifying both positive and negative instances. F1 score ranges from 0 to 1, where 1 represents perfect precision and recall, and 0 indicates the worst performance.

Matthews Correlation Coefficient (MCC): The MCC takes into account true positives, true negatives, false positives, and false negatives. It is a correlation coefficient between the predicted and actual classifications, considering all four aspects of the confusion matrix [40]. MCC ranges from -1 to +1, where +1 indicates a perfect prediction, 0 represents random guessing, and -1 signifies a complete disagreement between predictions and real values.

Area Under the Receiver Operating Characteristic Curve (AUC-ROC): The AUC-ROC metric measures the area under the receiver operating characteristic curve, which plots the true positive rate (recall) against the false positive rate. It provides a single value that represents the model's ability to distinguish between the two classes. AUC-ROC ranges from 0 to 1, where 1 indicates a perfect model.

These evaluation metrics enable us to comprehensively assess the performance of our CNN model in identifying the case column within the unbalanced dataset, considering both true positive and true negative predictions and addressing the challenges posed by class imbalance.

5.4 OVERVIEW OF THE STEPS TAKEN TO BUILD THE CNN MODEL

In the creation of our CNN model, we optimize both the model architecture and training hyperparameters. This approach naturally synchronizes these critical aspects of model development.

We focus on shaping the CNN's architecture and configuring essential training parameters, including the selection of a feature extraction network and hyperparameters like learning rate, batch size, and training epochs. Transfer learning, where a pre-trained model adapts to our specific task, expedites training. We also use the Optuna[1] to further enhance these hyperparameters.

Optuna plays a role in our CNN model development, enabling us to explore and determine the combination of both model architecture and training hyperparameters concurrently. This approach maximizes efficiency, saving time and resources while improving our model's performance.

By adopting this standard practice of simultaneous optimization, we ensure a coherent model development process that efficiently explores hyperparameter spaces.

5.5 DEFINITION OF THE MODEL ARCHITECTURE HYPERPARAMETERS AND TRAINING PROCESS HYPERPARAMETERS

In the construction of the CNN models, the focus is on defining the overall model architecture and training process hyperparameters. This involves configuring various components to ensure the model's effectiveness in capturing features and making accurate predictions for the target variable.

1 https://optuna.org/

For the architecture, we begin by explored different pretrained models, and conducted extensive experiments with multiple hyperparameters to optimize and improve the model's performance. Table 6 outlines the options considered:

Table 6: Model Architecture Layers Options.

Layer	**Values Tried**	**Always Present**
1 | Feature Extraction	VGG16 / ResNet50 / InceptionV3	X
2	Flatten / GlobalAveragePooling2D	X
3 | Dense Layer 1	Units: 2^X / Activation: ReLU / X: 1-9	
4 | Dropout 1	0.0-0.5	
5 | Dense Layer	Units: 1 / Activation: Sigmoid	X

Within Table 6, it is important to highlight that the Feature Extraction layer corresponds to the Feature Extraction Section, while the subsequent layers form the Classification Section in the model's structure.

The options considered were the following:

- **Feature Extraction section**: We explored three popular pre-trained models for feature extraction: VGG16, ResNet50, and InceptionV3. These models are renowned for their exceptional performance in image recognition tasks, and we assessed how they adapt to our specific problem.
- **Flatten / GlobalAveragePooling2D layers**: After feature extraction, we need to prepare the data for the classification layer. Two common approaches are to either flatten the final feature map or use Global Average Pooling. We evaluated both options to determine which one works best for our dataset.
- **Hidden Dense Layer**: This layer was designed to experiment with varying numbers of hidden units, specifically employing 2^X units, where X ranged from 1 to 9 after flattening or pooling. We applied the ReLU activation function, known for its effectiveness in promoting faster convergence and mitigating the risk of vanishing gradients.
- **Dropout Layer**: To prevent overfitting, we experimented with dropout rates ranging from 0.0 to 0.5. Dropout is a regularization technique that randomly drops a fraction of the neurons during training.
- **Output Dense Layer**: The output dense layer consists of a single unit with the sigmoid activation function. As this is a binary classification problem, the

sigmoid activation function squashes the output to a probability between 0 and 1, indicating the likelihood of the input image belonging to the positive class (case id).

In the training process hyperparameters optimizers are a crucial component. It determines how the model's weights are updated during the training process to minimize the loss function. We experimented with the following optimization techniques [41]

RMSprop: Root Mean Square Propagation is an adaptive learning rate optimization algorithm. It adjusts the learning rate for each weight based on the moving average of the squared gradient. RMSprop is suitable for recurrent neural networks and deep learning models.

Adam: Adam is another adaptive optimization algorithm that combines the benefits of RMSprop and stochastic gradient descent with momentum. It adapts the learning rate for each parameter based on estimates of the first and second moments of the gradients.

SGD: Stochastic Gradient Descent is a classic optimization algorithm for training neural networks. It updates the model's parameters in the direction of the negative gradient of the loss function.

Adadelta: Adadelta is an extension of Adagrad that addresses the diminishing learning rate problem. It dynamically adapts the learning rates of each parameter based on previous updates.

Nadam: Nadam combines Adam with Nesterov accelerated gradient (NAG) optimization. It incorporates NAG's momentum term with Adam's adaptive learning rates.

The learning rate is a critical hyperparameter that determines the step size at which the optimizer moves towards the minimum of the loss function. Choosing an appropriate learning rate is essential, as it influences the speed and stability of the model's training process. We experimented with different learning rates between 0.0001 to 0.1, to identify the optimal value for the generated model.

5.6 CNN RESULTS

In this section, we present the results obtained with the CNN model for the case identification task. We begin by showcasing the result obtained in each step.

Table 7: Model Architecture Hyperparameters.

Layer	Values
1 | Feature Extraction	VGG16
2	Flatten
3 | Dense Layer 1	Units: 512 / Activation: ReLU
4 | Dense Layer	Units: 1 / Activation: Sigmoid

Table 8: Train Hyperparameters.

Hyperparameter	Values
Optimizer	Nadam
Learning Rate	0.00019509360022646507

The best combination of Model Architecture Hyperparameters (Table 7) returned by Optuna was: VGG16 for feature extraction, a flattening layer, a ReLU-activated dense layer with 512 units, and a Sigmoid-activated final layer.

The best Training Hyperparameters (Table 8) returned by Optuna were: the Nadam optimizer and a learning rate of about 0.0002.

Table 9 presents a summary of key metrics, including accuracy, precision, recall, AUC, MCC, and F1 Score. These metrics provide a comprehensive evaluation of our model's performance.

After conducting cross-validation with Optuna, we observed significant discrepancies in performance metrics between train and test datasets across multiple folds, signifying the presence of overfitting. Our initial plan to fine-tune the models after cross-validation was abandoned due to the clear signs of overfitting.

Our exploration into the application of CNNs for our case identification task ultimately yielded outcomes that fall short of our desired objectives at this juncture. To address the challenge of overfitting, we outline a future research direction,

Table 9: Summary Metrics Table.

Metric	Value
Accuracy	0.965517241
Precision	0.666666667
Recall	0.6
AUC	0.941158563
MCC	0.63720459
F1 Score	0.633333333

involving the application of techniques such as data augmentation and dataset balancing, designed to enhance the resilience of our models by mitigating overfitting issues effectively.

6 A HEURISTIC FOR CASE ID COLUMN IDENTIFICATION

In this chapter, we introduce an approach for identifying the Case ID column in the event log based on a heuristic that emulates the hypothesis introduced in Chapter 4. We will begin by describing the heuristic and follow with the presentation of our results.

6.1 THE HEURISTIC

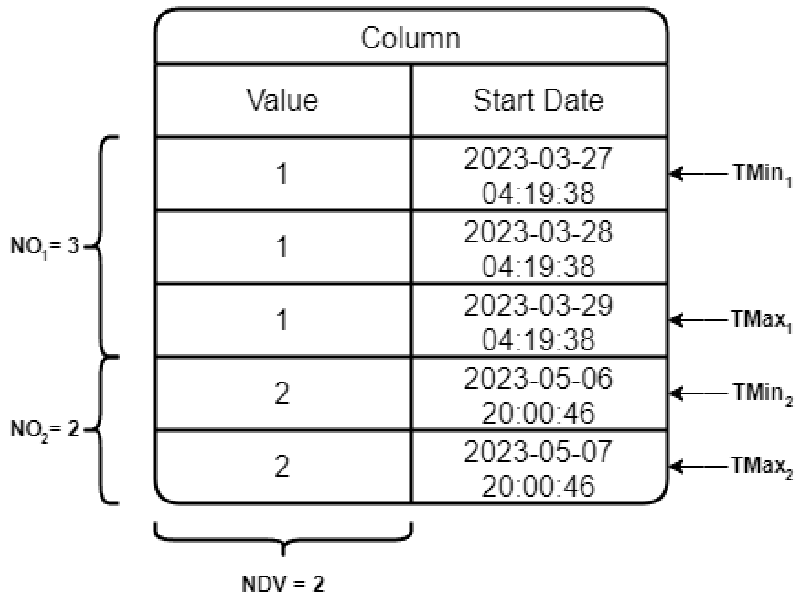
The proposed heuristic for identifying the “Case ID” column within event logs is based on the hypothesis, posed in Chapter 4, that *the “Case ID” column exhibits a shorter time span compared to other columns*. This means that the proposed heuristic needs to measure the average time span for the values of each column. Now, each column has different values and the number of occurrences of the values varies from value to value. That is, some values occur more often than others. So, instead of a simple average expression, we propose a heuristic consisting of a weighted average of values times spans where the time span for each value is weighted by the number of occurrences of that value.

By utilizing this weighted average, our heuristic effectively captures the temporal patterns that differentiate the “Case ID” column from others in the event log dataset.

The heuristic expression is defined as follows:

$$\text{Average time span} = \frac{\sum_i \left[(TMax_i - TMin_i)^a \times NO_i \right]^b}{NDV^c},$$

where T_{Max_i} is the maximum time value for a given value in the column, T_{Min_i} is the minimum time value for a given value in the column, NO_i is the number of occurrences a value has in the column, NDV is the number of unique values in the column, a, b, and c are coefficients that can be adjusted to fine-tune the



Column	
Value	Start Date
1	2023-03-27 04:19:38
1	2023-03-28 04:19:38
1	2023-03-29 04:19:38
2	2023-05-06 20:00:46
2	2023-05-07 20:00:46

Figure 15: Heuristic expression variables example.

heuristic performance with the idea of changing the importance of each element in the expression.

In Figure 15, we show an example of the expression applied to assess a column's potential as the Case ID. This example concerns a table of a column, including attributes "value" and "start date".

Within the "value" attribute, two values, 1 and 2, are observed, each with respective counts (3 and 2). We also note the minimum and maximum dates linked to these values.

The final step is applying the expression:

$$\text{Value} = \frac{[(TMax_1 - TMin_1) \times NO_1] + [(TMax_2 - TMin_2) \times NO_2]}{NDV}.$$

This involves replacing observed values for calculation. This yields a numeric score, serving as a practical indicator for selecting the Case ID column based on temporal patterns and value occurrences.

By applying this formula to each column in the event log, we can obtain a numeric score representing its potential as the "Case ID" column. The lower the score, the more likely it is that the column is the appropriate choice for the Case ID.

6.2 HEURISTICS RESULTS AND DISCUSSION

To explore the effectiveness of the heuristic, we conducted experiments using different combinations of values for coefficients *a*, *b*, and *c*. We created a table that includes eight distinct sets of coefficient values, each representing a different configuration of the expression. This table serves as a reference for evaluating the impact of each element's impact on the results.

Table 10: Heuristics Configurations.

Heuristic	**a**	**b**	**c**
A	1	1	1
B	1	1	2
C	2	1	1
D	1	2	1
E	1/2	1	2
F	1	1/2	2
G	1/2	1	1
H	1	1/2	1

Table 10 presents the eight tested configurations of the heuristic. Each configuration is characterized by a specific combination of values of coefficients *a*, *b*, and *c*.

Configuration A, with a=1, b=1, and c=1, provides a balanced approach in evaluating the Case ID column. Configuration B, with a=1, b=1, and c=2, places slightly more emphasis on the number of unique values in the column. Configuration C, with a=2, b=1, and c=1, gives greater weight to the range of time values. In configuration D, with a=1, b=2, and c=1, the emphasis shifts towards the frequency of occurrence joined with the range of the time values. Configuration E, with a=1/2, b=1, and c=2, introduces a lower weight for the range of time values and a higher weight for the number of unique values. Configuration F, with a=1, b=1/2, and c=2, prioritizes the range of time values and the frequency of occurrence. Configuration G, with a=1/2, b=1, and c=1, gives equal weight to the range of time values and the frequency of occurrence. Finally, configuration H, with a=1, b=1/2, and c=1, balances the range of time values and the number of unique values.

We applied the heuristic with the above configurations to assess its effectiveness in identifying the Case ID column.

Table 11 presents the event log alongside the corresponding position rank assigned to the Case ID column for each method. The ranking system enables the identification of the most suitable Case ID column, aiding users in making informed decisions tailored to their specific event log and requirements.

Table 11: Comparison of expression configurations on different event logs.

Event	Heuristics								
log	A	B	C	D	E	F	G	H	Columns
[20]	2	2	2	2	1	1	2	2	54
[8]	1	1	1	1	1	1	1	1	4
[21]	1	1	1	1	1	1	1	1	11
[22]	1	1	1	1	1	1	1	1	11
[23]	1	1	1	1	1	1	1	1	10
[24]	1	1	2	2	1	1	1	1	17
[25]	2	1	3	3	1	1	2	2	16
[26]	1	1	1	1	1	1	1	2	16
[27]	1	1	1	1	1	1	1	2	17
[28]	1	1	1	1	1	2	1	2	18
[29]	1	1	1	1	1	2	1	2	15
[30]	3(2)	3(2)	3(2)	3(2)	3(2)	3(2)	3(2)	3(2)	27
[31]	1	1	1	1	1	1	1	1	14
[32]	1	1	1	1	1	1	1	1	5
[33]	1	1	1	1	1	1	1	1	17
[34]	1	1	1	1	1	1	1	1	17
[35]	1	1	1	1	1	1	1	1	9
[36]	1	1	1	1	1	1	1	3	58
Mould	2	4	2	2	4	4	4	4	15
Total	15/19	16/19	14/19	14/19	17/19	15/19	15/19	10/19	
Success Rate	79%	84%	74%	74%	89%	79%	79%	53%	

The table displays the event logs along with the corresponding rank obtained by the Case ID column for each method. For example, in the [20] event log, Configuration A resulted in a rank of 2, while Method E obtained a rank of 1, and so on. Similarly, for the [8] event log, all heuristics achieved a rank of 1. In some ranks we have a number between parenthesis, which corresponds to the number of columns that are on that rank, including the "Case ID" column.

The "Columns" column represents the number of columns in each event log that were considered for Case ID identification. This count includes all the potential candidate columns evaluated by the different heuristics. The number in this column provides an understanding of the event log's complexity and the variety of columns that were assessed during the Case ID identification process.

Furthermore, the "Total" row in the table summarizes the overall performance of each method across all event logs. The fractions indicate the number of event logs for which each method achieved the top rank out of the total number of event logs used in the experiments.

The analysis of the results from the various configurations of the expression provides insights into the effectiveness of different criteria in identifying the Case ID column.

In Table 11, it becomes evident that configuration E consistently outperformed other configurations by achieving the highest rank in the majority of event logs. This performance across different event logs suggests that configuration E is suitable for Case ID identification tasks.

On the other hand, Configuration H displayed relatively lower success rates. This variance in performance indicates that certain configurations may be more specialized and effective in specific scenarios, while others demonstrate broader adaptability.

An observation from these results is the significance of considering both the number of unique values (NDV) and the time span (T_{Max_i}-T_{Min_i}) when formulating the expression. Configuration E, which gives added weight to these factors, appears to excel in capturing the unique characteristics of the Case ID column.

In the project context, configurations A, C, and D were the top performers for identifying the Case ID in the mould event log. Importantly, these configurations do not give extra weight to the number of unique values (NDV). This stands in contrast to Configuration E, which was previously discussed but performed less effectively.

In order to provide a global perspective, it is worth noting that Configuration E achieved the top rank in approximately 89% of the event logs, making it the most consistently successful method. Additionally, it is remarkable that the Case ID column never ranked beyond the 4$^{\text{th}}$ position, and this only occurred for the mould event log, under various configurations, emphasizing the challenge posed by this specific event log.

The choice of configurations for the expression can be virtually infinite, and their performance may heavily depend on the specific characteristics of the event log. Different areas or domains within the event log may benefit from unique configurations tailored to their particular attributes. Additionally, the maturity or complexity of the event log can play a crucial role in determining which configurations of the expression yield the best results. Therefore, the adaptability of the expression to various event log scenarios underscores its versatility and potential for effective Case ID identification.

In percentage terms, the success rates across configurations vary as follows: Configuration H exhibited the lowest rate at 53%, followed by configurations C and D, jointly demonstrating a 74% success rate. Configurations A, G and F achieved a combined rate of 79%, while configurations B achieved 84%. Configuration E stood out with 89% success rate. These percentages provide a comprehensive overview of each configuration's performance across the event logs.

In [30], we consistently observed an identical ranking across all configurations. The first position was shared by the "docid" and "docid_uuid" columns, while the Case ID column and "identity:id" column jointly held the third position. This ranking might seem unusual, given the aggregation of columns with similar correlation values. However, both of them presented a correlation of around 0.99 that prevented their aggregation.

Comparing our approach to existing literature, specifically [13], [9] and [17], we emphasize the importance of assessing comparable scenarios. [13], utilizing the BPI 2013 event log, involved data manipulation, including selective case ID selection and the removal of cycling events. In contrast, our approach primarily centres on column elimination without data manipulation, making a direct comparison unfeasible.

The study in [9] covered multiple datasets, including [8], [22], and [29]. Interestingly, our method closely aligns with theirs in BPI 2012 and 2013, accurately selecting the case ID column. However, a notable deviation becomes apparent in the BPI 2017 dataset, where our approach exhibits performance discrepancies across all heuristic configurations. While we share similarities in two out of the three datasets, the discrepancies in the third dataset underscore the intricacies of case ID identification across diverse event logs.

The work in [17], on the other hand, focused on the [8] and [22] datasets but employed a different evaluation approach. Instead of direct case ID identification, they concentrated on precision and recall calculations for the generated event log. This variance in evaluation metrics makes direct comparisons unfeasible.

7 SOFTWARE LIBRARY DEVELOPMENT

In this chapter, we discuss the creation of a software library to identify the "Case ID" column in unlabelled event logs. Its main goal is to streamline column selection, making the process more efficient for users in data analysis, process mining, and related fields.

The library's audience includes individuals and professionals in data analysis, process mining, and related fields. Its final purpose is to assist users in identifying the "Case ID" column in an event log-based business process dataset.

The software library primarily focuses on deducing relevant columns like "Start Date" and "End Date", with a key emphasis on identifying the "Case ID" column. It also allows users to select the event column and generate XES files, a standard format for process mining. Additionally, it offers functionality to calculate performance metrics related to event data, providing insights into process efficiency.

The subsequent sections of this chapter dive into crucial aspects of the software library development. First, we explore its architecture and design, outlining the modular structure presented as a Python class. Next, we cover the implementation, detailing the coding approach and algorithms used to achieve the library's functionalities. Following that, we delve into the testing section, describing comprehensive testing methods ensuring the accuracy and reliability of the library's outputs. These sections aim to provide a comprehensive view of the library's development process, offering users a strong foundation for leveraging its capabilities.

7.1 DEVELOPMENT ENVIRONMENT

The development setup centres on Docker, using the Python 3.10.6-alpine base image. Docker offers an efficient and consistent environment for software library development, ensuring easy replication across various systems.

To enable the library's functionality along with necessary dependencies, the Docker environment includes add-ons supporting the cuDF and pandas libraries.

This setup harnesses the processing speed of cuDF and pandas for data manipulation within the container.

Using Docker with the Python 3.10.6-alpine base image provides flexibility in testing the library with different Python versions. Modifying the specified Python version in the Docker configuration allows straightforward validation of the library's compatibility with other Python releases.

Additionally, Docker containerization isolates the development environment, containing the library's dependencies within the container and preventing interference with the host system. This isolation mitigates version conflicts and dependency issues, leading to a smoother development process.

Overall, Docker adoption simplifies the library's development, testing, and deployment workflows, enhancing maintainability and scalability across various Python environments.

7.2 ARCHITECTURE AND DESIGN

In this section, we explore the architecture and design of our software library. The key principle guiding the design of this library is modularity. A modular architecture brings several advantages. By breaking down the functionality into self-contained and independent modules, we achieve enhanced flexibility, reusability, and maintainability.

Each module in the library serves a specific purpose, such as deducing start and end dates, identifying the "Case ID" column, selecting the event column, generating XES files, and calculating performance metrics (see Figure 16). The modular design enables users to interact with specific functionalities independently, making it easy to use only the modules relevant to their specific needs.

Furthermore, the library provides an object-oriented interface, where users create an instance of the library, which serves as a variable to access various functions while retaining the state across function calls. This design paradigm allows users to maintain and manage the context of their operations efficiently. For instance, they can initiate an object with a specific event log, and then use the same object to deduce dates, identify Case IDs, and perform other process mining tasks without losing track of the underlying data and configurations.

The modular architecture improves readability and comprehensibility. Each module encapsulates a specific functionality, making the codebase more organized and

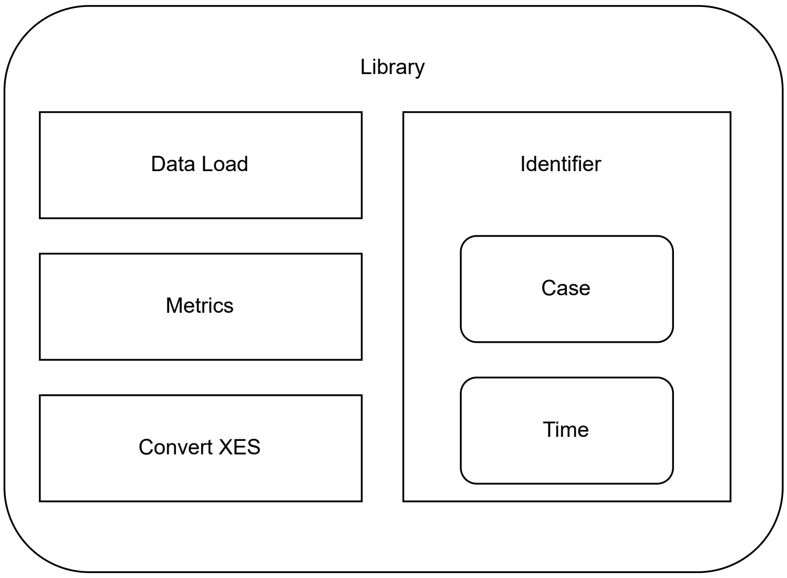


Figure 16: Architecture of the Software Library.

understandable. It promotes code reuse, as the same module can be used across different projects or scenarios, thereby streamlining the development process.

The design of this software library also facilitates extensibility. Users can easily integrate new functionalities or enhancements without affecting the existing codebase. They can develop separate modules for distinct tasks and seamlessly integrate them into the library.

In addition to its modular design and object-oriented interface, it is essential to emphasize that the library mandates the use of NVIDIA graphics cards with CUDA technology for GPU processing.

7.3 DEVELOPMENT

The library's core purpose is to identify crucial components within unlabelled event logs, specifically the case identifier and time-related information. We provide automated solutions to streamline data handling, extraction, and transformation processes, catering to data analysts and process mining enthusiasts. The library also offers performance metrics for evaluation and facilitates the conversion of processed data into standardized formats for compatibility with process mining tools, making data analysis more accessible and efficient for users across diverse domains. For documentation of the library, refer to Appendix A.

7.3.1 *Data Load*

The implementation phase of the software library development involves creating specific modules to achieve the library's objectives. One of the initial modules developed during this phase was the Data Load module, which serves the crucial role of efficiently loading data from the file provided by the user.

The objective of the Data Load module is to provide a mechanism for ingesting data into the library. It handles CSV files and uses the pandas library, known for its data manipulation and analysis capabilities, making it adaptable to different dataset structures.

The Data Load module is designed to simplify the user experience while ensuring adaptability. When users provide the file location as input, the data loader automatically detects the separator character for CSV files, eliminating the need for manual specification. This automated detection enables smooth data ingestion, reducing the burden on users and minimizing input errors.

In addition to its automatic separator detection, the Data Load module implements error handling mechanisms. In cases where users encounter file-related issues or provide unsupported formats, the module provides feedback and error messages. By communicating the specific nature of the error, users can identify and resolve problems, enhancing the usability of the library.

7.3.2 *Identifier*

In this section, we delve into the intricacies of the Identifier module, which plays a pivotal role in automatically identifying and extracting essential information from the event log.

This module comprises two distinct components: the Time module and the Case module. Each of these modules focuses on extracting specific columns from the event log that are crucial for time-related analysis and case identification, respectively.

The Time module uses the same systematic approach employed in Chapter 4 to examine the event log's columns and identify those containing date-related information. Once the Time module has successfully detected the date columns, it proceeds to ascertain the temporal boundaries. It determines the oldest date in the event log as the starting point (Start Date) for the process mining analysis. Simultaneously, the module identifies the newest date as the end point (End Date).

These boundaries define the time window within which the event log data operates, encapsulating the temporal scope of the process instances.

By automatically identifying the temporal boundaries through a systematic exploration of the event log, the Time module lays the groundwork for further temporal analysis and process mining tasks. The precise and efficient identification of the time window empowers users to conduct in-depth investigations into the duration, frequency, and patterns of process activities, providing valuable insights for process improvement and optimization.

Continuing with the Identifier module, we explore the Case module, which includes the proposed heuristic in Chapter 6, which can be configured using parameters: a, b, and c. The main goal of the Case module is to pinpoint the event log's case identifier column, a critical step for future process mining and analysis.

Subsequently, the module applies the method with different configurations selected by the user and creates a ranking for each one based on their individual performance. If the user opts to select only one configuration, the module automatically designates the column associated with the top-ranked configuration as the case identifier. This ranking mechanism provides valuable insights into the efficacy of each configuration, helping users to determine the most suitable approach for their event log. By considering the rankings, users can confidently identify the optimal configuration for case identification or, if desired, combine multiple methods to create a robust and versatile case identification process.

Furthermore, the Time and Case modules offer users the flexibility to make manual selections based on their preferences and domain knowledge, with the possibility of selecting between CPU or GPU processing. If the user prefers to control the process more explicitly, they can choose specific columns for the time and case identification purposes. This manual selection option allows users to tailor the analysis to their specific event log characteristics, enabling a more customized and informed approach to the identification of temporal information and case identifiers.

7.3.3 *Metrics*

In the Metrics module, the library offers functionality to calculate the execution time taken by each of the individual modules. This includes measuring the execution time for the Data Load module, Time module, Case module, and any other future modules that may be added to the library. By providing detailed time measurements for each module, users gain insights into the performance and efficiency of their data processing pipeline.

Moreover, the Metrics module also computes the processing time for each column in the event log. This feature allows users to identify potential bottlenecks and assess the impact of individual columns on the overall data processing time. By understanding the time distribution across columns, users can make informed decisions about data preprocessing, optimize column selection, and improve the efficiency of their data analysis workflow.

7.3.4 *Convert XES*

This module is a critical part of the library, converting processed event logs into the XES file format, commonly used in process mining. It enables users to integrate their data with process mining tools.

During execution, the module takes the processed event log, including the identified case identifier column and time information, and generates an XES file. This file captures temporal event patterns, suitable for various process mining tasks like discovering models and measuring metrics.

The module offers customization options, allowing users to adapt the output XES file to their needs. Users can choose columns as attributes, define classifiers, and specify extension attributes, aligning the file with their specific process mining workflow.

7.4 TESTING

In the development of our software library, a crucial aspect was the testing performed to ensure its reliability and functionality. This involved manual testing, which played a pivotal role in identifying potential issues and refining the library's performance.

During the manual testing phase, all functions within the library were subjected to comprehensive examination, encompassing a wide range of input scenarios. This encompassed both anticipated and unanticipated inputs, allowing us to assess how the library responded to diverse conditions.

Additionally, we thoroughly evaluated the library's handling of valid and invalid inputs, which was a vital aspect of the testing process. This step helped us identify vulnerabilities and areas where the library might be prone to errors.

When issues or discrepancies within the library surfaced during manual testing, we addressed them. This included making necessary modifications and enhancements to enhance the library's stability and reliability.

The objective of manual testing was to ensure that the software library could perform effectively in real-world scenarios, delivering robust functionality and dependable performance.

8 CONCLUSION AND FUTURE WORK

This study encompassed data collection, data understanding, data preparation, the development of a Convolutional Neural Network model and a heuristic for Case ID column identification, and the creation of a software library.

In the data collection phase, we obtained 19 event logs that correspond to our dataset, which underwent data understanding and preparation. In the data understanding, two hypotheses were made: first, the diagonal pattern within event logs and, second, the expectation that the Case ID column should exhibit a smaller average time span compared to other columns.

The data preparation was done in two steps. First, we employed various techniques to reduce the number of columns that were not important. These techniques were: remove data format columns, remove empty columns, remove columns with two unique values and correlation-based column aggregation. Second for making it ready for the heuristic and the CNN model we have done: data filtering, null value exclusion, data sorting and categorical transformation.

For Case ID column identification, we employed two approaches: a Convolutional Neural Network model and a heuristic method.

For CNN model creation, we started with the creation of the image dataset that was generated from each column after the data preparation phase. However, this approach resulted in an unbalanced dataset, with only 19 Case ID images compared to 332 images for not Case ID columns. During the model architecture hyperparameters training process using cross-validation, we observed that the models were exhibiting symptoms of overfitting. This led us to pause the model development and reconsider the planned fine-tuning process.

A heuristic was created based on the mean average but tailored to address our specific hypotheses. This heuristic expression can be adjusted using the exponents to assign varying importance to each of them. We conducted tests with eight different configurations, and the best configuration successfully identified the Case ID column in 17 out of 19 event logs, corresponding to 89% success rate. Furthermore, 5 of the 8 tested configurations yielded a success rate of 79% or more.

This work focused on creating a library for Case ID column identification. The library, with a modular architecture, includes the previous heuristic method and a time column detection as its main features, enhancing its practicality for this purpose. Additionally, this library finds application in decision support systems, where user verification is essential to ensure its effectiveness.

For researchers and business professionals engaged in process mining, the developed library presents an opportunity to delve into uncharted territories of exploration and analysis. It intends to provide an accessible and efficient solution, simplifying the comprehension of intricate event log data, thus facilitating deeper insights and informed decision-making.

8.1 FUTURE WORK

One avenue for future work involves expanding data preparation capabilities. Specifically, we can explore the addition of a mechanism to exclude more columns from the event log. This enhancement would refine input data, potentially improving performance of the expression-based approach and address overfitting in the CNN model. By allowing users to selectively exclude irrelevant columns, we can fine-tune data input for more accurate Case ID detection.

One avenue for future work is to improve the CNN model by addressing overfitting issues. To achieve this, we plan to re-implement the CNN model with a focus on data augmentation and dataset balancing. By augmenting the dataset with variations of existing samples and ensuring a balanced representation of classes, we aim to create a more robust model capable of handling diverse input data effectively.

An intriguing path for future research is exploring alternative deep learning architectures for Case ID column detection. Investigate the use of Siamese networks, instead of the CNN, for syntactic analysis. A Siamese network could be employed to leverage syntactic analysis for improved performance. A comparative study between the CNN and a Siamese network could provide insights into the advantages and disadvantages of each approach.

For future library development, we aim to investigate activity column detection, closely linked to Case ID column detection in process mining. Our goal is to create a dedicated module for activity column detection, expanding the toolkit for event log preprocessing. This addition will enhance the library's usefulness for process mining and data analysis. Additionally, we plan to integrate the library into Python-based process mining tools like PM4Py to assess its usability through user testing.

These four key areas — enhanced data preparation, data augmentation and dataset balancing, exploring Siamese networks, and investigating activity column detection — present opportunities to advance Case ID column detection in unlabelled event logs and test the library's usability for related tasks in process mining.

BIBLIOGRAPHY

[1] W. M. P. van der Aalst, "Process mining: A 360 degree overview," in *Process Mining Handbook*, W. M. P. van der Aalst and J. Carmona, Eds. Cham: Springer International Publishing, 2022, pp. 3–34, ISBN: 978-3-031-08848-3. DOI: 10.1007/978-3-031-08848-3_1. [Online]. Available: https://doi.org/10.1007/978-3-031-08848-3_1.

[2] M. Gupta and P. Chandra, "A comprehensive survey of data mining," *International Journal of Information Technology*, pp. 1–15, Feb. 2020. DOI: 10.1007/s41870-020-00427-7.

[3] D. Amyot, O. Akhigbe, M. Baslyman, *et al.*, "Combining goal modelling with business process modelling: Two decades of experience with the user requirements notation standard," *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 17, pp. 2–1, 2022.

[4] Wu, He, Wang, Wen, and Yu, "A business process analysis methodology based on process mining for complaint handling service processes," *Applied Sciences*, vol. 9, no. 16, p. 3313, Aug. 2019, ISSN: 2076-3417. DOI: 10.3390/app9163313. [Online]. Available: http://dx.doi.org/10.3390/app9163313.

[5] F. S. Angelo Corallo Mariangela Lazoi, *Process mining and industrial applications: A systematic literature review*. John Wiley and Sons Ltd, 2020.

[6] Á. Rebuge, "Business process analysis in healthcare environments," Ph.D. dissertation, May 2012.

[7] W. Van Der Aalst, *Process mining: data science in action*. Springer, 2016, vol. 2.

[8] B. van Dongen, *Bpi challenge 2012*, 2012. DOI: 10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F. [Online]. Available: https://data.4tu.nl/articles/_/12689204/1.

[9] A. A. Andaloussi, A. Burattin, and B. Weber, "Toward an automated labeling of event log attributes," vol. 318, Springer Verlag, 2018, pp. 82–96, ISBN: 9783319917030. DOI: 10.1007/978-3-319-91704-7_6.

[10] N. Martin, *Data Quality in Process Mining*. 2021, pp. 53–79. DOI: 10.1007/978-3-030-53993-1_5.

[11] M. T. Wynn and S. Sadiq, *Responsible Process Mining - A Data Quality Perspective*. Springer Verlag, 2019, vol. 11675 LNCS, pp. 10–15, ISBN: 9783030266189. DOI: 10.1007/978-3-030-26619-6_2.

[12] F. Folino, G. Folino, M. Guarascio, and L. Pontieri, "Learning effective neural nets for outcome prediction from partially labelled log data," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, pp. 1396–1400. DOI: 10.1109/ICTAI.2019.00196.

[13] M. Sahu, G. K. Nayak, and R. K. Nayak, "Process model discovery from unlabeled event logs by using non-overlapping sequential distinct event patterns," *International Journal of Engineering Research and Technology*, vol. 13, pp. 3055–3066, 10 2020, ISSN: 09743154. DOI: 10.37624/IJERT/13.10.2020.3055-3066.

[14] M. Walicki and D. R. Ferreira, "Sequence partitioning for process mining with unlabeled event logs," *Data and Knowledge Engineering*, vol. 70, pp. 821–841, 10 Oct. 2011, ISSN: 0169023X. DOI: 10.1016/j.datak.2011.05.003.

[15] T. Lichtenstein, D. Bano, and M. Weske, "Attribute-driven case notion discovery for unlabeled event logs," in Aug. 2022, pp. 111–122, ISBN: 978-3-030-94342-4. DOI: 10.1007/978-3-030-94343-1_9.

[16] D. Bayomie, I. M. Helal, A. Awad, E. Ezat, and A. ElBastawissi, "Deducing case ids for unlabeled event logs," vol. 256, Springer Verlag, 2016, pp. 242–254, ISBN: 9783319428864. DOI: 10.1007/978-3-319-42887-1_20.

[17] D. Bayomie, A. Awad, and E. Ezat, "Correlating unlabeled events from cyclic business processes execution," Aug. 2016, pp. 274–289, ISBN: 978-3-319-39695-8. DOI: 10.1007/978-3-319-39696-5_17.

[18] I. M. A. Helal, A. Awad, and A. E. Bastawissi, "Runtime deduction of case id for unlabeled business process execution events," 2015, pp. 1–8. DOI: 10.1109/AICCSA.2015.7507132.

[19] S. Sim, R. A. Sutrisnowati, S. Won, S. Lee, and H. Bae, "Automatic conversion of event data to event logs using cnn and event density embedding," *IEEE Access*, vol. 10, pp. 15 994–16 009, 2022. DOI: 10.1109/ACCESS.2022.3143609.

[20] B. van Dongen, *Real-life event logs - hospital log*, 2011. DOI: 10.4121/UUID:D9769F3D-0AB0-4FB8-803B-0D1120FFCF54. [Online]. Available: https://data.4tu.nl/articles/_/12716513/1.

[21] W. Steeman, *Bpi challenge 2013, closed problems*, 2013. DOI: 10.4121/UUID:C2C3B154-AB26-4B31-A0E8-8F2350DDAC11. [Online]. Available: https://data.4tu.nl/articles/_/12714476/1.

[22] W. Steeman, *Bpi challenge 2013, incidents*, 2013. DOI: 10.4121/UUID:500573E6-ACCC-4B0C-9576-AA5468B10CEE. [Online]. Available: https://data.4tu.nl/articles/_/12693914/1.

[23] W. Steeman, *Bpi challenge 2013, open problems*, 2013. DOI: 10.4121/UUID:3537C19D-6C64-4B1D-815D-915AB0E479DA. [Online]. Available: https://data.4tu.nl/articles/_/12688556/1.

[24] B. van Dongen, *Bpi challenge 2015 municipality 1*, 2015. DOI: 10.4121/UUID:A0ADDFDA-2044-4541-A450-FDCC9FE16D17. [Online]. Available: https://data.4tu.nl/articles/_/12709154/1.

[25] B. van Dongen, *Bpi challenge 2015 municipality 2*, 2015. DOI: 10.4121/UUID:63A8435A-077D-4ECE-97CD-2C76D394D99C. [Online]. Available: https://data.4tu.nl/articles/_/12697349/1.

[26] B. van Dongen, *Bpi challenge 2015 municipality 3*, 2015. DOI: 10.4121/UUID:ED445CDD-27D5-4D77-A1F7-59FE7360CFBE. [Online]. Available: https://data.4tu.nl/articles/_/12718370/1.

[27] B. van Dongen, *Bpi challenge 2015 municipality 4*, 2015. DOI: 10.4121/UUID:679B11CF-47CD-459E-A6DE-9CA614E25985. [Online]. Available: https://data.4tu.nl/articles/_/12697898/1.

[28] B. van Dongen, *Bpi challenge 2015 municipality 5*, 2015. DOI: 10.4121/UUID:B32C6FE5-F212-4286-9774-58DD53511CF8. [Online]. Available: https://data.4tu.nl/articles/_/12713285/1.

[29] B. van Dongen, *Bpi challenge 2017*, 2017. DOI: 10.4121/UUID:5F3067DF-F10B-45DA-B98B-86AE4C7A310B. [Online]. Available: https://data.4tu.nl/articles/_/12696884/1.

[30] B. van Dongen and F. (Borchert, *Bpi challenge 2018*, 2018. DOI: 10.4121/UUID:3301445F-95E8-4FF0-98A4-901F1F204972. [Online]. Available: https://data.4tu.nl/articles/_/12688355/1.

[31] B. van Dongen, *Bpi challenge 2019*, 2019. DOI: 10.4121/UUID:D06AFF4B-79F0-45E6-8EC8-E19730C248F1. [Online]. Available: https://data.4tu.nl/articles/_/12715853/1.

[32] B. van Dongen, *Bpi challenge 2020: Domestic declarations*, 2020. DOI: 10.4121/UUID:3F422315-ED9D-4882-891F-E180B5B4FEB5. [Online]. Available: https://data.4tu.nl/articles/_/12692543/1.

[33] B. van Dongen, *Bpi challenge 2020: International declarations*, 2020. DOI: 10.4121/UUID:2BBF8F6A-FC50-48EB-AA9E-C4EA5EF7E8C5. [Online]. Available: https://data.4tu.nl/articles/_/12687374/1.

[34] B. van Dongen, *Bpi challenge 2020: Prepaid travel costs*, 2020. DOI: 10.4121/UUID:5D2FE5E1-F91F-4A3B-AD9B-9E4126870165. [Online]. Available: https://data.4tu.nl/articles/_/12696722/1.

[35] B. van Dongen, *Bpi challenge 2020: Request for payment*, 2020. DOI: 10.4121/UUID:895B26FB-6F25-46EB-9E48-0DCA26FCD030. [Online]. Available: https://data.4tu.nl/articles/_/12706886/1.

[36] B. van Dongen, *Bpi challenge 2020: Travel permit data*, 2020. DOI: 10.4121/UUID:EA03D361-A7CD-4F5E-83D8-5FBDF0362550. [Online]. Available: https://data.4tu.nl/articles/_/12718178/1.

[37] [Online]. Available: https://www.ibm.com/topics/neural-networks.

[38] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and video compression with neural networks: A review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1683–1698, 2020. DOI: 10.1109/TCSVT.2019.2910119.

[39] D. Bhatt, C. Patel, H. Talsania, *et al.*, "Cnn variants for computer vision: History, architecture, application, challenges and future scope," *Electronics*, vol. 10, no. 20, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10202470. [Online]. Available: https://www.mdpi.com/2079-9292/10/20/2470.

[40] M. Heydarian, T. E. Doyle, and R. Samavi, "Mlcm: Multi-label confusion matrix," *IEEE Access*, vol. 10, pp. 19 083–19 095, 2022. DOI: 10.1109/ACCESS.2022.3151048.

[41] E. M. Dogo, O. J. Afolabi, and B. Twala, "On the relative impact of optimizers on convolutional neural networks with varying depth and width for image classification," *Applied Sciences*, vol. 12, no. 23, p. 11 976, Nov. 2022, ISSN: 2076-3417. DOI: 10.3390/app122311976. [Online]. Available: http://dx.doi.org/10.3390/app122311976.

APPENDICES

A

APPENDIX A: LIBRARY DOCUMENTATION

A.1 MODULE NAME

`ProcessMiningForDummies`

A.2 DESCRIPTION

This module provides a class that enables the identification of Case ID columns and their corresponding date columns in a CSV file containing event log data. It offers a range of functions and utilities for efficient preprocessing and analysis of event log data, essential for process mining and data-driven decision-making.

A.3 FUNCTIONS

- `__init_(file_path: str) -> None`
  - Constructor for the `ProcessMiningForDummies` class.
  - **Parameters**:
    - `file_path` (`str`): The file path to the CSV dataset containing event log data.
  - **Returns**:
    - None
  - **Example**:

    ```
    import CaseIdIdentification
    variable = CaseIdIdentification.ProcessMiningForDummies('bpi_2011.csv')
    ```
- `get_predicted_dates_columns() -> Tuple[List[str], str, str]`
  - Predicts and retrieves date columns and their temporal range from the event log data.

  - **Returns** a tuple containing:
    - `dates_columns` (`List[str]`): A list of identified date columns.
    - `start_date` (`str`): The start date column.
    - `end_date` (`str`): The end date column, if applicable.
  - **Example**:

    `variable.get_predicted_date_columns()`

- `set_date_columns(start_date: str, end_date: str = None) -> None`
  - Manually sets the start and end date columns for temporal analysis.
  - **Parameters**:
    - `start_date` (`str`): The start date column.
    - `end_date` (`str`, optional): The end date column, if applicable.
  - **Returns**:
    - None
  - **Example**:

    `variable.set_date_columns('startDate','endDate')`

- `get_predicted_case_id_configurations() -> List[str]`
  - Retrieves a list of available configurations for predicting the Case ID column in the event log data.
  - **Returns**:
    - `methods` (`List[str]`): A list of available configurations for Case ID prediction of this work.
  - **Example**:

    `methods = variable.get_predicted_case_id_methods()`

- `get_Option_List() -> List[str]`
  - Retrieves a list of options available for configuring the Case ID prediction process.
  - **Returns**:
    - `options` (`List[str]`): A list of available options for Case ID prediction.
  - **Example**:

```
options = variable.get_Option_List()
```

- `getOptionExponent(option: str) -> float`
  - Retrieves the exponent associated with a specific option for Case ID prediction. The exponent is used to configure the prediction process.
  - **Parameters**:
    * `option` (`str`): The option for which to retrieve the exponent.
  - **Returns**:
    * `exponent` (`float`): The exponent value associated with the specified option.
  - **Example**:

```
exponent = variable.getOptionExponent('option_name')
```

- `get_predicted_case_id_column(configurations: List[str] = [], tmp: bool = True, gpu: bool = False, folder: str = None, performance: bool = False) -> List[Dict[str, Union[str, Dict[str, float]]]]`
  - Predicts the Case ID column in the event log data using specified methods and configurations.
  - **Parameters**:
    * `method` (`List[str]`, optional): A list of methods to use for prediction.
    * `tmp` (`bool`, optional): Indicates whether to use temporary files for prediction.
    * `gpu` (`bool`, optional): Indicates whether to use GPU for prediction.
    * `folder` (`str`, optional): The folder path for storing prediction results.
    * `performance` (`bool`, optional): Indicates whether to collect performance metrics during prediction.
  - **Returns**:
    * `ranking` (`List[Dict[str, Union[str, Dict[str, float]]]]`): A list of dictionaries containing ranking results for Case ID prediction methods.
  - **Example**:

```
ranking = variable.get_predicted_case_id_column(method=[
{'a':1,'b':1,'c':1}], cnn=True, performance=True)
```

- `set_case_id_column(case_id_column: str) -> None`
  - Sets the Case ID column for further analysis.
  - **Parameters**:
    - `case_id_column` (`str`): The name of the Case ID column.
  - **Returns**:
    - None
  - **Example**:

    ```
    variable.set_case_id_column('CaseID')
    ```
- `set_activity_column(activity_column: str) -> None`
  - Sets the Activity column for further analysis.
  - **Parameters**:
    - `activity_column` (`str`): The name of the Activity column.
  - **Returns**:
    - None
  - **Example**:

    ```
    variable.set_activity_column('Activity')
    ```
- `convert_xes(file_path: str = None, file_name: str = ’log.xes’) -> str`
  - Converts the event log data to the XES format and saves it to a specified file.
  - **Parameters**:
    - `file_path` (`str`, optional): The folder path where the XES file will be saved.
    - `file_name` (`str`, optional): The name of the XES file.
  - **Returns**:
    - `xes_file_path` (`str`): The path to the saved XES file.
  - **Example**:

    ```
    variable.convert_xes(file_path='/output', file_name='event_log.xes')
    ```

- `get_metrics() -> Dict[str, Any]`
  - Retrieves performance metrics collected during the Case ID prediction process.
  - **Returns**:
    - `metrics` (`Dict[str, Any]`): A dictionary containing performance metrics.
  - **Example**:

    ```
    metrics = variable.get_metrics()
    ```

DECLARAÇÃO

Declaro, sob compromisso de honra, que o trabalho apresentado neste projeto, com o título *“Case ID Detection in Unlabelled Event Logs for Process Mining”*, é original e foi realizado por André Alexandre dos Santos Vicente (2210791) sob orientação de: Rui Pedro Charters Lopes Rijo, Ricardo Filipe Gonçalves Martinho e Carlos Fernando de Almeida Grilo.

Leiria, setembro de 2023

Assinado por: **ANDRÉ ALEXANDRE DOS SANTOS VICENTE**
Num. de Identificação: BI14660738
Data: 2023.12.08 22.11.12 GMT Standard Time

CHAVE MÓVEL

André Alexandre dos Santos Vicente