Online Process Extractor and Updater

Sónia Rafaela Costa da Rocha

Master's degree in Network and Information Systems Engineering
Department of Computer Science

2022

Orientador
Ricardo Teixeira Sousa, Advisor to the Centre Coordinator, Laboratory of Artificial Intelligence and Decision Support at INESC TEC

Sworn Statement

I, Sónia Rafaela Costa da Rocha, enrolled in the Master Degree in Network and Information Systems Engineering at the Faculty of Sciences of the University of Porto hereby declare, in accordance with the provisions of paragraph a) of Article 14 of the Code of Ethical Conduct of the University of Porto, that the content of this dissertation reflects perspectives, research work and my own interpretations at the time of its submission.

By submitting this dissertation, I also declare that it contains the results of my own research work and contributions that have not been previously submitted to this or any other institution.

I further declare that all references to other authors fully comply with the rules of attribution and are referenced in the text by citation and identified in the bibliographic references section. This dissertation does not include any content whose reproduction is protected by copyright laws.

I am aware that the practice of plagiarism and self-plagiarism constitute a form of academic offense.

Sónia Rocha,

December 16th, 2022.

Abstract

With the growth of the data volume, managing such information becomes very difficult in today's systems due to the amount of files they contain. Processes define how an institution organizes its work and resources, such as the order in which operations are done and which groups of individuals are permitted to perform certain tasks.

Process mining tools make a big impact in the quality of service of an industry since it supports the maintenance and organization of the management of a process. Process mining intends to improve the existing processes, through cost reduction, management time reduction and error removal.

Process Management Systems emerged from the result of the necessity for a structured and organized information environment. These systems need tools not just for extracting processes, but also for updating, monitoring, and improving them.

This thesis objective was to develop a tool capable of extracting and updating process models from event logs in an online scenario.

With the collaboration of IPBrick who delivered the data, it was possible to create the online tool that not only creates process models but assists in choosing the best algorithm for the specified process by evaluating the results of the replay fitness, precision, simplicity, generalization and soundness. In this application there will be the possibility to see the results given by the models. In order to retrieve the model for the IPBrick database, a Petri-net Markup language file was possible to download from the tool.

Resumo

Com o crescimento do volume de dados, torna-se muito difícil nos sistemas atuais gerir a quantidade de arquivos que eles contêm. Os processos definem como uma organização organiza seu trabalho e recursos, como a ordem em que as operações são feitas e quais grupos de indivíduos têm permissão para executar determinadas tarefas.

As ferramentas de process mining têm grande impacto na qualidade de serviço de uma indústria, pois ajudam a manter uma estrutura organizada entre a gestão dos processos. Process mining pretende melhorar os processos existentes, através da redução de custos, redução de tempo de gestão e eliminação de erros.

Os Sistemas de Gestão de Processos surgiram como resultado da necessidade de um ambiente de informação estruturado e organizado. Esses sistemas precisam de ferramentas não apenas para extrair processos, mas também para atualizá-los, monitorá-los e melhorá-los.

O objetivo desta tese foi desenvolver uma ferramenta capaz de extrair e atualizar o modelo de processo existente, apartir de dados de eventos, em um cenário online.

Com a colaboração da IPBrick que forneceu os dados, foi possível criar a ferramenta online que não só cria modelos de processos como auxilia na escolha do melhor algoritmo para o processo especificado, avaliando os resultados de replay fitness, precisão, simplicidade, generalização e soundness. Nesta aplicação haverá a possibilidade de ver os resultados dados pelos modelos. Para retornar o modelo para a base de dados da IPBrick, foi possível descarregar da ferramenta um ficheiro do tipo Petri-net Markup Language.

Agradecimentos

Este projeto marca o fim de uma grande jornada de cinco anos na Faculdade de Ciências da Universidade do Porto e todo apoio que recebi tornou esta jornada ainda mais marcante.

Por estes anos todos tenho a agradecer pelas memórias, os conselhos, ajudas e apoio emocional dadas pelo Simão, o Ângelo e o Eduardo, foram sem dúvida as pessoas que mais me marcaram nos meus anos de faculdade.

Quero agradecer aos meus pais, avós e à minha irmã pelo suporte incondicional que me deram e as palavras de coragem que foram fundamentais para nunca deixar de acreditar em mim.

Um agradecimento grande ao meu orientador Ricardo Sousa, pela ajuda, cuidado e orientação dados para a finalização desta etapa.

Gostaria também de agradecer ao professor Raul Oliveira da IPBrick pois a sua ajuda foi crucial para a parte experimental deste projeto.

Por fim, quero agradecer a toda a gente que fez parte destes anos que passei na faculdade pois todos tiveram uma influencia positiva no meu trajeto.

Acknowledgements

I made contributions to the project with the reference **NORTE-01-0247-FEDER-069218**. All the data present in this work was provided by **IPBrick**.

Contents

List of Tables

List of Figures

Listings

Acronyms

BPM Business Process Management

XES Extensible Event Stream

XML Extensible Markup Language

FCUP Faculty of Sciences of the University of Porto

IM Inductive Miner

IMd Inductive Miner Directly-Follows

IMf Inductive Miner Infrequent

LIAAD Laboratory of Artificial Intelligence and Decision Support

PNML Petri Net Markup Language

PM4Py Process Mining for Python

WFN Workflow-Net

Chapter 1

Introduction

1.1 Context and Motivation

Nowadays the volume of information is increasing exponentially, therefore more documents and files are present on today's systems. This high rate of documents makes it harder for companies to manage them, therefore it became necessary to find means to rapidly and effectively manage the workflows of the processes that exist in the organizations [3].

Within companies, there has been a transition to a process focus environment instead of the previous data focus. Processes translate how an organization assembles its work and resources, such as the sequence in which activities are completed and which groups of people are authorized to undertake specific jobs [36].

Organizations may have highly precise process definitions of how work is organized, which are assisted by a process aware information system such as a workflow management system [36].

In a range of application fields, such as healthcare, education, and software development, process mining has been effectively applied to gather insight understanding of business processes [3].

Process mining tools make a big impact in the quality of service of an industry, since they maintain an organize environment between the management of processes. Process mining information may be also utilized to improve existing processes, for example, through cost reduction, management time reduction, and error removal.

Process Management Systems emerged from the need to have a structured and organized information environment. This systems require tools not only to extract processes but to update, monitor and improve them. In process documentation, the existing actual processes are recorded, and process models are typically made to store and illustrate this information.

This thesis objective was to develop an online tool that would extract processes from events logs and update the existing process model in an online scenario. This work was made possible

thanks to the collaboration of Laboratory of Artificial Intelligence and Decision Support (LIAAD) department of INESC TEC, Faculty of Sciences of the University of Porto (FCUP) and IPBrick. This tool was very beneficial to IPBrick since it allowed the gathering of safe, reliable, automatic and less complex models, saving time and interventions of specialists.

Access to the iPortalDoc platform from IPBrick was essential, this platform is a Document Management System where it is possible to create manually workflows from a process. The processes present in this platform were the ones used to test the developed application.

1.2 Problem definition

Nowadays the implementation of a document and process management system is essential to the good functioning of a company, they give the possibility to manage documents and processes as well as document and archive information in a organized way.

Process mining aims to automatically construct a process model that represents the behavior recorded in an event log. Process mining models can be used as a starting point for the deployment of technologies that assist process execution and/or as a feedback mechanism to compare the supplied process model with the implemented one [32].

In document management systems where the process model is created manually, a lot of limitations appear since the user is required to have prior knowledge of the processes, therefore it becomes time consuming and prone to human errors.

Based on the importance of creating an online tool to automatically extract processes in organizations, the current project intends to perform process extraction and possibility update the resulted model. The big question in this project was: Can we automatically extract a model that is reliable and well-aligned with the processes?

The data recorded by a process mining tool may be utilized to gain a better understanding of the actual processes, such as analyzing discrepancies and improving the model quality.

The growth of event data in organizations cause a bigger number of devices recorded therefore, as event logs grow, process mining techniques need to become more efficient and highly scalable [5]. Processes and information must be completely coordinated to fulfill compliance, efficiency, and customer service criteria [10].

Event data, in the form of logs, is essential to the beginning of a process discovery operation resulting in a process model. In order to make this model reliable, it will be important to test several algorithms with the intent to have the best output for document process description. The main goal was not only to develop the online tool but to understand the impact it makes in the industry.

1.3 Methodology

The primary goal of this thesis was the development of a tool capable of extracting processes from event logs creating a process model. The secondary goal was to make the most reliable process model that describes the intended process, by comparing the different process discovery algorithms. The process discovery technique was the technology behind the methodology.

The data was provided by IPBrick, a company specialized in Corporate Communication Solutions with its core product being a robust Communication Platform. Their primary areas of expertise are: Document and Process Management, Enterprise Social Network, Unified Communications and Mail&Groupware.

Within the several proposed technologies within IPBrick, iPortalDoc is the most relevant in the context of this thesis. This software is a document management and workflow system for companies and institutions [27]. iPortalDoc was mainly used to compare the created process models with the workflows provided by this platform with the purpose to better understand the structure of the process.

For the implementation of the process mining techniques Python programming language was used, considering that is the preferred for prototyping in this field. The API PM4PY package was crucial since it supports several process mining useful tools such as Extensible Event Stream (XES) files importing/exporting, petri net compatibility, functions for discovery algorithms and several more useful tools.

There were other Python libraries that were essential in this work such as the Panda package with the purpose to, make analysis and manipulation of data, Graphviz since it supported the visualization of structural information with graphs and networks, Streamlit which provided tools to make the online platform and much more. All the information about the development phase is explained in Chapter 3.

1.4 Dissertation Structure

This dissertation is organized in four more chapters. The Chapter 2 dives into the State of The Art of Process Mining. In Chapter 3 focuses in the development phase and the steps behind it. The discussion of the results is described in Chapter 4. Finally, Chapter 5 breaks down the conclusions of the thesis and possible future work.

Chapter 2

State of The Art

2.1 Introduction

The main focus of the State of The Art chapter will be on the process mining subject. This chapter, will start by covering an important domain that process mining could be applied to, document process management (Section 2.2). After that, it will be explored the definition of Business Process Management (Section 2.3) and the important relation it has with the beginning of the process mining technology.

After that, we will dive into the different process mining techniques: discovery (Section 2.4.1), conformance (Section 2.4.2) and enhancement (Section 2.4.3).

Near the ending Sections, the methodological challenges in this field will be mentioned (Section 2.5), the most used tools will be stated (Section 2.6), and finally a synthesis of the whole State of The Art (Section 2.7).

To collect such information keyword-based searches for conference papers and articles were done in databases such as Engineering Village and Mendeley. Some of the keywords used were: ”process mining”, ”process discovery”, ”process models”, ”event logs” and ”document and process management”.

2.2 Document Process Management

Document Management Systems provides means for a structured and centralized management of information, avoiding losses of time and information waste.

Over time the amount of documents in organizations has been increasing, making it harder to manage them. Documents have a lot of parameters that need to be taken into account in order to have a structured environment. For example, the need to categorize a document, to know the version its on, its destination and if its critical information, are only a few of the many

other parameters in a document.

Taking that into consideration, it becomes easier to understand that without a document and management system, the circulation of external or internal documentation, as well as the processing of information, is time-consuming and error prone. These mistakes frequently result in document loss and, as a result, process inefficiencies [28].

A Document and Process Management System like iPortalDoc can solve this issue, since it can trigger, for example, a workflow just by sending an email [28]. Although, as mentioned in the Section 1.2, if a process model is created manually (as in iPortalDoc), it comes with a lot of limitations, such as, a higher probability of human error, time consumption and the need to have prior knowledge of the processes.

Process mining (Section 2.4) would be a great solution for this type of issues, since, with this technology, it is possible to automatically create a process model. In order to document processes, the amount of detail at which they are being recorded must be determined. This is established by the process strategy and the resulting process architecture. Good documentation provides the analysis of processes, allowing bottlenecks and further improvement opportunities to be identified.

2.3 Business Process Management

Business Process Management (BPM) is a field that encompasses methods for designing, executing, controlling, measuring, and optimizing business processes [6]. Process mining did not arise from data mining or machine learning contrary to what others might suppose. Process mining has its origins in the subject of BPM [7]. Observing the BPM life-cycle is the greatest way to understand the connection between process mining and BPM [6].

The life-cycle shown in the Figure 2.1 has several phases that represent the management of a business process [3]:

- in the *design* phase it occurs the design of a process;
- if the system supports the designed process then the *enactment/monitoring* phase begins. In this step, processes will be tracked in order to detect if any changes are required;
- this modifications will be handled by the *adjustment* phase. The process is not rebuilt at this phase, and no new software is developed; only preset controls are utilized to adapt or reconfigure the process;
- finally, the *diagnosis/requirements* phase identifies processes and detects any new additional requirements which could set off a new iteration in the cycle starting with the *redesign* phase.

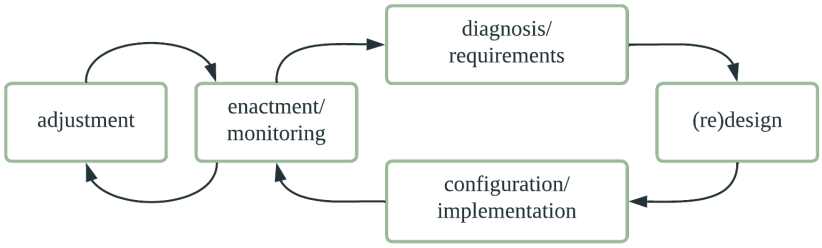

Figure 2.1: BPM life-cyle (Adapted from: [6]).

Process mining offers a way to optimize the BPM life-cycle, that currently, in most organizations, only severe issues or critical external changes will prompt a new iteration of the life-cycle, therefore the true knowledge about the present process is rarely used in the redesign phase [6].

Process mining enables the possibility to complete the BPM life-cycle. Data collected by information systems may be utilized to gain a better understanding of real processes, such as analyzing discrepancies and improving the model quality. In process mining the goal is to discover, monitor and optimize real processes by extracting information stored in an event log [6].

2.4 Process Mining

The primary goal of process mining is to automatically create a process model that represents the behavior in an event log.

It is considered that events may be recorded in such a way that each event: refers to an activity, refers to a case (i.e. process instance), can have information about the person who executed or initiated the activity, must include a timestamp and is completely ordered [36].

There are three main perspectives in process mining [2]:

- The *control-flow perspective* focuses on the order at which activities are performed. The purpose of this perspective is to find a feasible characterization of all possible pathways, expressed in a modelling notation such as Petri nets.
- The *organizational perspective* focuses on the resources concealed in the log, aiming to structure an organization, for example, by classifying people based on their functions.
- The *case perspective* is concerned with the properties of cases. Cases can be classified based on where they are in the process or who is working on them.

There are three types of process mining techniques: discovery, conformance and enhancement. Respectively, there will be a description of each in the next subsections, (2.4.1, 2.4.2, 2.4.3) . This thesis will focus on process discovery, which is the process of extracting a model from an event log describing the control-flow.

2.4.1 Process Mining Discovery

Discovery is the first step of any process mining procedure. Without utilizing any a-priori knowledge, a discovery approach takes an event log and generates a model [8] .

Organizations use processes to manage cases. However, in most situations, processes are informal and may not have been documented at all. Furthermore, even when processes are documented, reality may be extremely different. As a result, it is critical to use event data to uncover the real processes [4].

The discovered process models may be used for a variety of reasons, including problem discussion among stakeholders, creating process improvement ideas and model enhancement [4].

Process mining is concerned with the analysis, verification, and enhancement of process models in workflow systems. The analysis of these logs (process discovery) employs a plethora of algorithms (see Section 2.4.1.4) in an attempt to expose the structure and links between the various events in the logs. In the next Section (2.4.1.1) event logs and its characteristics will be clarified.

The Section 2.4.1.2 will cover some of the challenges that process discovery faces. In the Section 2.4.1.3 some of the most used process model notations will be described. Finally in the Section 2.4.1.5 the four process model quality dimensions will be explained.

2.4.1.1 Event Log

An event log is a database of event sequences used in process discovery. Each event records a change in the state of an activity of a certain type in real time [24]. The standard format for event logs is Extensible Event Stream (XES). The IEEE Task Force on Process Mining has selected the standard as the primary exchange format for event logs.

Process mining aims to extract information about a specific process from event logs. Typically, these techniques presume that it is feasible to record events in a sequential manner, with each event referring to an activity and being associated with a specific instance. Additionally, some mining approaches make use of extra parameters such as the event's originator, the event's timestamp, or other items recorded with the event [2].

In order to perform process mining techniques there is a minimum of three variables that an event log must contain [19]:

- **Activity**: the different process actions or status modifications that were executed in the process. For example, for an employee in order to book its vacation, it may take several stages: the request, the verification and the acceptance.
- **Case ID**: one or more columns that, when combined, uniquely identify a particular process execution, constituting the Case ID. This parameter represents the several instances of a

process, i.e., the process steps. The case ID the identifier of a specific activity.

- **Timestamp**: indicates when each of the activities occurred. It is used to, not only analyze the process's time behavior, but also for establishing the order of the activities in the event log.

2.4.1.2 Challenges

Process discovery still has a lot of challenges when generating a process model from event logs. Some of the most challenging problems are [25]:

- **Noise**: Process mining algorithms presume that the information given is reliable. Although in most cases this is a reasonable assumption, the log may contain information that was logged improperly, i.e., "noise". The mining algorithm must be resistant to noise, which means that causal relations shouldn't be established on the basis of only one observation. Process discovery methods must avoid overfitting this noise as a result [9].
- **Incompleteness**: A log is considered incomplete if it does not have sufficient information to derive the process [9]. The concept of completeness is also highly essential in process mining. It has a relation with noise, however, while noise describes the problem of having "too much data" (describing unusual behavior), completeness describes the problem of having "too little data" [3].
- **Prior knowledge**: In a real-world scenario, the actual process is usually unknown a priori. Therefore, it is challenging to determine whether it led to a justifiable mining outcome. Justifiability refers to the degree to which an induced model is consistent with the existing domain knowledge.
- **Duplicate activities**: The issue with duplicate tasks refers to the scenario in which two nodes in a process model (e.g., a Petri net) relate to the same task. Since the duplicate activities have the same "pattern" in the log, most algorithms combine them into a single action, producing a inaccurate or counter-intuitive model [2].
- **Balance between "overfitting" and "underfitting"**: Process mining algorithms must achieve a balance between "overfitting" and "underfitting". Overfitting occurs when a model fails to generalize and only allows for the exact behavior recorded in the log. An underfitting model overgeneralizes what is shown in the log, allowing for more behavior to be seen even though there are no signs in the log that this additional behavior is possible [2].

2.4.1.3 Process Model Notations

In order to represent the behavior present in a event log a modeling notation must be chosen, there exist several proposed notations. In this thesis we are going to take a closer look into Petri

Nets, Workflow-nets and Process trees which are three of the most used modeling notations in this field:

- **Petri Net**: Petri nets are a formal semantics-based graphical language that has been used to depict, analyze, verify, and simulate dynamic behavior. Petri nets are composed of places, tokens, and arcs [25].

  Places (shown as circles), Figure 2.2, are a distributed representation of state that can include tokens. Each alternate distribution of tokens throughout the places of a Petri net represents a different state, referred to as a marking.

  Transitions (represented by rectangles), Figure 2.2, can consume and create tokens while also representing a state change. Usually, the transitions of the resulted model correspond to the activities in the event log.

  Arcs (drawn as arrows), Figure 2.2, link points and illustrate a flow connection.

  Definition 2.4.1 (Petri net [26])**.** $PN = (P,\ T;\ F)$ is a Petri net when:

  - P and T are a finite set of places and transitions respectively; and
  - $F \subseteq (T \times P) \cup (P \times T)$ is a set of directed arcs connecting places and transitions.

  In Petri Nets, $\forall t \in T$, $\bullet t = \{p|\ (p,t) \in F\}$ is referred to the input set and $t\bullet = \{p|\ (t,p) \in F\}$ is referred to the output [26]. A place p is designated an input place of a transition t if, and only if, there is a directed arc from p to t, also, a place p is designated an output place of transition t if, and only if, there is a directed arc from t to p [1].

  The firing rule determines how a Petri net behaves. A transition t is activated if each input place p of t holds at least one token. When a transition is set, it can fire. When a transition fires, the Petri net's state changes.

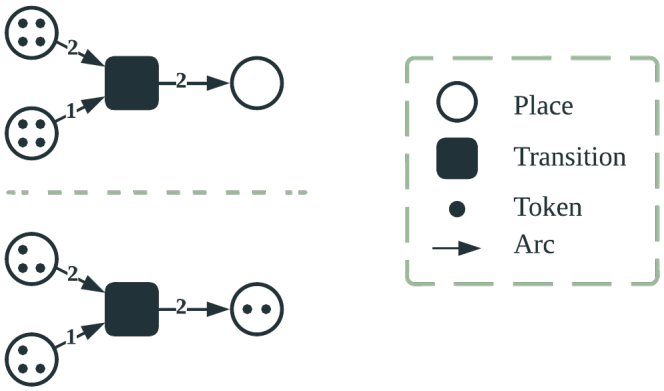

Figure 2.2: Transition firing example (Adapted from: [21]).

Fundamentally, as can be seen in the Figure 2.2 one token from each input place p is consumed and creates one token from each output place p of the transition t.

Petri nets may depict structures such as OR/AND splits and OR/AND joins as can be seen in the Figure 2.3.

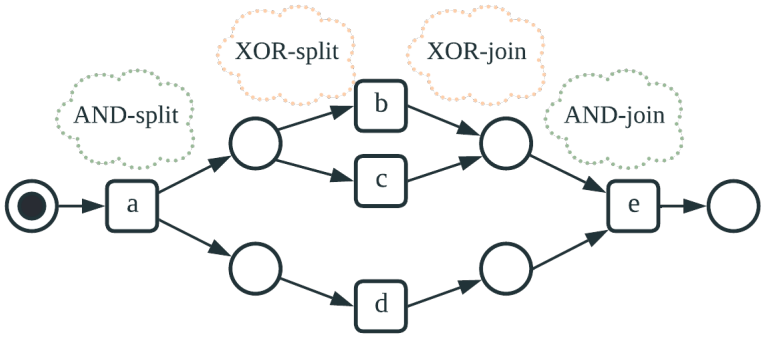

Figure 2.3: AND/JOIN splits and OR/AND joins structures in a Petri net (Adapted from: [3]).

- **Workflow-Net**: A Workflow-Net (WFN) is a Petri net with a single source and sink point that models the beginning and end state of a process.

  Definition 2.4.2 (WFN [26]). *WFN* = (*PN*, *i*, *o*) is a WFN when *PN* = (*P*, *T*; *F*) is a Petri net and with input place *i* and output place *o*:

  – if there is a single source place $i \in P$, and $\bullet i = \emptyset$;
  – if there is a single sink place $o \in P$ and $o\bullet = \emptyset$; and
  – when $\forall x \in P \cup T$, x is on a path from i to o.

  A WFN contains four fundamental structures, these being sequential, parallel, choice, and loop. An example of these structures is demonstrated in the Figure 2.4.

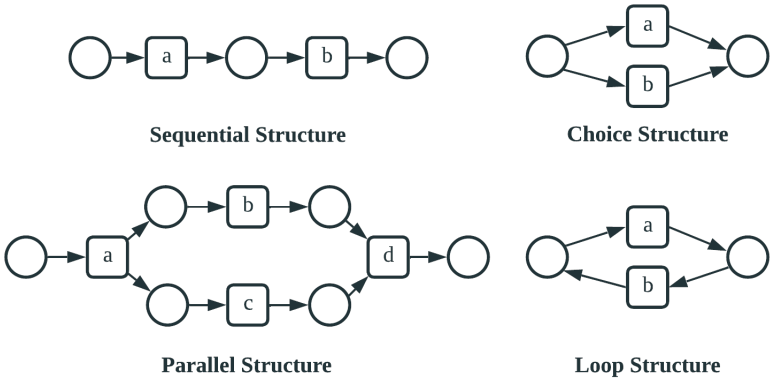

Figure 2.4: Four fundamental structures of a WFN [Adapted from: [26]]

Not every WFN indicates a valid process. A process defined by a WFN, for example, may display faults such as deadlocks, activities that never become active, livelocks, or waste remaining in the process after termination. As a result, we define the well-known correctness criteria [3]:

Definition 2.4.3 (Soundness [3]). Let *WFN* = (PN,i,o) with input place *i*, output place *o* and *PN* = (*P*, *T*; *F*) being a Petri net. WFN is sound if and only if:

- *Safeness*: $(N, [i])$ is safe, i.e., in a the WFN model, each place can only hold a single token.
- *Proper completion*: If the sink place is marked in the WFN model, then all other places must be and remain empty, i.e., when a process is finished, none of its events can be executed.
- *Option to complete*: It must always be feasible to end the process. This indicates that the sink place must always be possible to be marked in the WFN model.
- *Absence of dead parts*: $(N, [i])$ contains no dead transitions. Indicating that in the WFN model, for any transition $t \in T$ there must be a firing sequence enabling t.

- **Process tree**: A process tree is a directed connected graph that does not have any cycles. In the graph, a node is either a branch node or a leaf node. Each leaf node represents an activity from the process's collection of activities. Each branch node, also known as the operator node, contains one or more children [15].

  The labeling function assigns an operator to each operator node and an activity to each leaf node. The currently defined operators are sequence, exclusive choice, parallel, or, and loop constructions. All operators require at least two branches, except for the loop operator, which accepts just one branch [15].

  Each of the operators has portions that may be translated to other well-known process modeling notations. The Figure 2.5 illustrates each operator's translation from a Process tree to a Petri net.

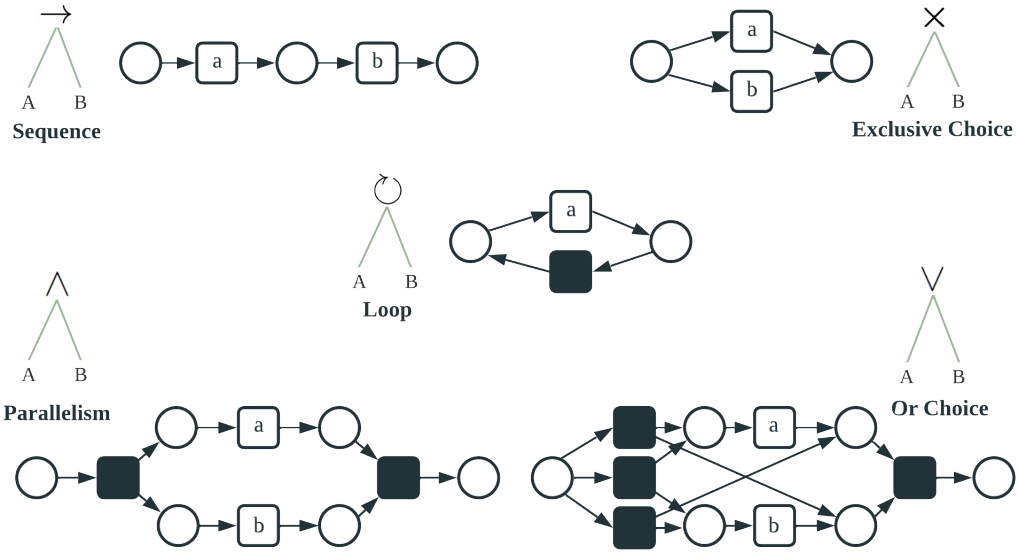

Figure 2.5: Operator's translation from a Process tree to a Petri net (Adapted from: [15]).

2.4.1.4 Algorithms

In this section we will cover three of the most known algorithms regarding process mining discovery techniques, named: Alpha Miner, Heuristic Miner and Inductive Miner.

- **Alpha Miner**: The process mining research community introduced the alpha algorithm as the first process discovery algorithm. The algorithm's core idea is to uncover the correlations between each event and then create a process model based on these interactions.

  It is a straightforward approach that begins with concurrency. The alpha algorithm searches the event log for certain patterns. For example, if an activity a is followed by b but b is never followed by a, it is considered that a and b have a causal dependency [4].

  The algorithm defines four different types of relationships: sequence, choice, parallel, and loop. Although the alpha method is simple to grasp, it has several flaws. Its main weakness is that it does not use frequencies which makes it weak against noise. It is only suited for event logs with no noise, which is relatively uncommon in learning data [13]. This method can't detect small loops.

  Many variants, such as the alpha+, alpha++, and alpha*, have been proposed to address some of the drawbacks. These variants additionally investigate event ordering relationships and build a more reliable Petri net as a process model [33]. However, alpha miner and its variants can't ensure soundness.

- **Heuristics Miner**: The heuristic algorithm uses the frequency of events and sequences to build a process model analogous to a causal net [33].

  The algorithm first constructs a dependency graph based on the activities frequencies and the number of times one action is followed by another. Dependencies are added to the graph based on preset thresholds (or not).

  The dependency graph displays the process model's "skeleton". This skeleton is used to explore the split and join behavior in detail. If an activity has numerous input/output arcs, the heuristic miner examines the log to determine if the join is AND, XOR, or OR [4].

  This method offers three major advantages over the alpha algorithm: it considers frequencies and significance to filter out noisy or unusual behavior, making it less susceptible to noise and incomplete logs; it is capable of detecting small loops and it allows for the omission of individual activities. However, it does not ensure soundness [13].

- **Inductive Miner**: the inductive algorithm is a method for constructing a process tree from a given log. The primary benefits of this algorithm are [23]:

  - All detected models match to sound, block-structured workflow net systems.
  - The model always fits, i.e., by introducing unseen transitions, the inductive algorithm follows the principles to discover a model with perfect fitness.

The solution employs a recursive divide and conquer strategy: split log, build part of the process tree, and proceed with processing the split log sections independently. A process tree may be turned into a Petri Net if necessary. To build process models, the Inductive miner makes use of the same type of relationships between nodes, i.e., sequence, choice, parallel, and loop [33].

2.4.1.5 Process Model Quality Dimensions

Process models should result in a high-quality model with the least complexity possible, providing sufficient accuracy, recall, precision, and generalization [22]. As shown in the Figure 2.6 there are four quality dimensions that are used to assess how effectively a process model describes the actual behavior seen in a log:

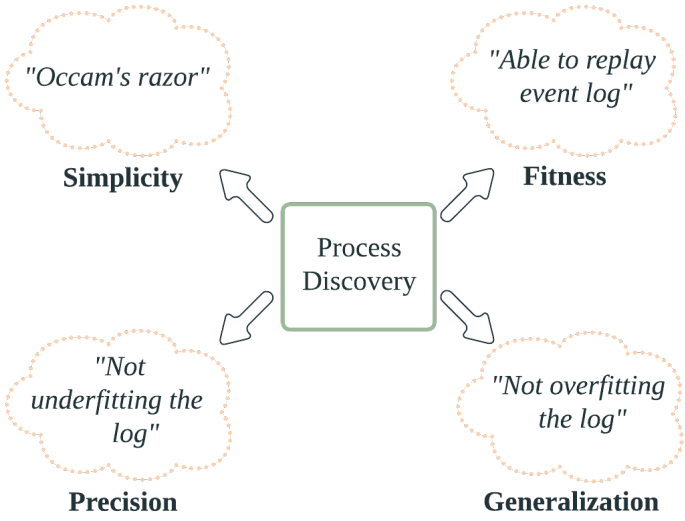

Figure 2.6: Quality dimensions in process model discovery (Adapted from: [8]).

- **Simplicity**: measures how easy it is for a human to interpret the process model. In the Figure 2.6 there is a reference to the Occam's Razor principle, which states that "one should not increase, beyond what is necessary, the number of entities required to explain anything" [8], i.e., the less complex the created model is, the easier it will be to understand it. There are different measures to classify the simplicity of a process model, although, the size being the main complexity criterion [14]. The class of alpha-algorithms is a type of process discovery algorithms that focuses on simplicity. These strategies often produce simplistic models with low replay fitness and/or precision [16].
- **Replay Fitness**: specifies the fraction of the activity in the event log that the process model can replay, i.e., the extent to which the discovered model can faithfully reproduce the cases recorded in the log. There are several measures for this dimension, some more detailed then others. The most current and reliable method employs a cost-based technique

between the traces in the event log and the process model's most optimum execution, by changing the costs, additional flexibility and differentiation is provided between more and less important tasks [14]. When finding a process model, certain algorithms, such as the heuristics miner, employ replay fitness as a guiding principle, however this does not guarantee ideal results [16].

- **Precision**: is linked with the concept of underfitting that is presented in data mining. A model with a low precision underfits the observed behavior in the log [3].
- **Generalization**: is associated with the concept of overfitting that is presented in data mining. An overfitting model does not generalize sufficiently, implying that it is excessively specialized and heavily influenced by examples from the event log [3].

2.4.2 Process Mining Conformance Checking

Conformance checking compares an event log to a process model with the goal to identify deviations and gather diagnostic information [11].

The expressiveness of a process modelling language has a direct impact on the simplicity of models and analytical capabilities, therefore, conformance checking approaches are often based on process models expressed as Petri nets or Process trees. [30].

As an input, conformance checking approaches need an event log and a technical process description (process model). Event data must be related to their associated process instance, therefore, there is a clear consensus on the event log architecture. The minimum content requirements are a unique case identifier, an activity label, and a time-stamp [18].

Furthermore, event logs may provide supplemental information on different process viewpoints, such as resources, costs, and so on, which conformance checking algorithms can use for further analysis [18].

Deviations are created by process executions that do not correspond to the process model and are typically linked with longer throughput times and poor quality levels. As a result, it is critical to recognize them, understand their origins, and re-engineer the process to avoid such deviations [11].

Conformance checking can be used for a wide range of purposes, including evaluating the reliability of documented processes, identifying deviating cases and understanding what they have in common, judging the quality of a discovered process model, and serving as a starting point for model enhancement [4].

The ability to replay the event log on the model is required for both conformance checking and performance analysis. Various replay mechanisms, such as token-based [11] replay and alignments [11], have been proposed.

2.4.3 Process Mining Enhancement

In process mining the event log may also be used to extend or enhance an existing process model. A non-fitting process model can be adjusted utilizing the diagnostics offered by the model and log alignment [4].

There are several proposed methods for finding bottlenecks, for example, by monitoring the time difference between casually related events and computing basic statistics like averages, variances, and confidence intervals [4].

Following the discovery of the process model, the conformance checking method can be used to identify the similarities and differences between the modeled behavior and observed behavior; i. e., the conformance checking technique determines whether the process model accurately reflects the actual process in the recorded event log as mentioned in the Section 2.4.2. If the process model does not accurately reflect reality, the goal is either to repair or extend the model so that they can replay the majority of the event log [37].

There are two types of process enhancement, extend and repair. Extend intends to adding additional perspectives to the model using event data. Repair entails improving the quality of the model using event data [37].

2.5 Methodological Challenges

Despite the application of process mining, many obstacles persist; these demonstrate that process mining is an evolving discipline. Here are some of the most significant methodological challenges, defined in [10], that this technology still encounters:

i) Finding, Merging, and Cleaning Event Data: Several issues must be handled when extracting event data suited for process mining, such as, data may be scattered across several sources, event data may be incomplete, an event log could contain outliers, logs may contain events at different levels of granularity, and so on.

ii) Dealing with Complex Event Logs Having Diverse Characteristics: Event logs can have a variety of characteristics. Some event logs may be quite vast, making them difficult to manage, whilst others may be so little that there is insufficient data to draw valid conclusions.

iii) Creating Representative Benchmarks: Because process mining is a emerging discipline, it requires suitable benchmarks, that are still missing, such as example data sets and representative quality standards to evaluate and enhance the various tools and algorithms.

iv) Dealing with Concept Drift: The process may change while being analyzed, this occurrence is referred to ”concept drift”. Understanding this issue is critical for process management. As a result, further study and tool assistance are required to fully investigate

concept drift.

v) Improving the Representational Bias Used for Process Discovery: A process discovery approach generates a model using a specific language. The selection of a target language frequently includes multiple implicit assumptions. It constrains the search space; processes that cannot be expressed in the target language are not detected. To provide high-quality process mining outcomes, the "representational bias" must be chosen with greater attention and refinement.

vi) Balancing Between Quality Criteria such as Fitness, Simplicity, Precision, and Generalization: There are four competing quality dimensions: fitness, simplicity, precision, and generalization as explained in the Section 2.4.1.5. The main problem is to identify models that perform well in all four dimensions.

vii) Cross-Organizational Mining: Process mining has generally been used inside a single organization. However, as service technology, supply-chain integration, and cloud computing grow more prevalent, there are circumstances in which many companies' event logs are available for analysis.

In "cross-organizational" process mining, there are two options: some firms collaborate to manage process instances (for example, supply chain partners), whereas others execute the same process while sharing experiences, expertise, or a infrastructure.

For both forms of "cross-organizational" process mining, new analysis methodologies must be created. These approaches should also take into account privacy and security concerns. As a result, developing privacy-preserving process mining approaches is critical.

viii) Providing Operational Support: Nowadays many data sources are updated in (near) real-time and there is enough computational capacity available to evaluate events as they occur. Therefore, process mining should not just be utilized for offline analysis; it may also be used for online operational support.

In an online setting, when a case deviates from the predefined process, the system can recognize it and produce an alert. Using process mining techniques in such an online environment adds new obstacles in terms of computational power and data quality.

ix) Combining Process Mining With Other Types of Analysis: It is challenging to gain additional insights from event data by integrating automated process mining methods with other analytical approaches (optimization techniques, data mining, simulation, visual analytics, etc.).

x) Improving Usability for Non-Experts: Process mining aims to construct process models that are used on a regular basis. As a result, it is critical and difficult to conceal advanced process mining algorithms behind a user-friendly interface that automatically selects parameters and recommends appropriate types of analysis.

xi) Improving Understandability for Non-Experts: The user may struggle to comprehend the output or be inclined to draw wrong conclusions. To prevent such issues, the results should be provided in an appropriate representation, with the reliability of the results always explicitly emphasized.

2.6 Tools

The area of process mining includes tools and techniques for increasing overall understanding of a process by evaluating event data stored throughout the process's execution [12]. Some tools are free to use and open source, such as ProM, RapidProM and Apromore. Commercial technologies include Fluxicon's Disco, Celonis, Perceptive Process Mining, Aris Express, Comprehend (OpenConnect), Process Discovery Focus (iontas), Enterprise Visualization Suite (Businesscape), BPM One (Pallas Athena) and QPR ProcessAnalyzer.

ProM is the most widely used tool in this industry. ProM is an open source toolkit created at the Eindhoven University of Technology in the Netherlands. ProM is an excellent choice for investigating process mining because it has continually been up to date in this technology.

This software is a framework that can be extended to accommodate a wide range of process mining techniques via plug-ins. It is platform independent because it is written in Java and may be downloaded for free.

ProM includes a number of plug-ins that allow the user to apply the most recent advances in process mining research to their own data. In the Figure 2.7 is shown an example of a Petri net result, with a plugin for the inductive miner algorithm that offers an interactive visualization for process exploration, performance analysis and deviation detection.

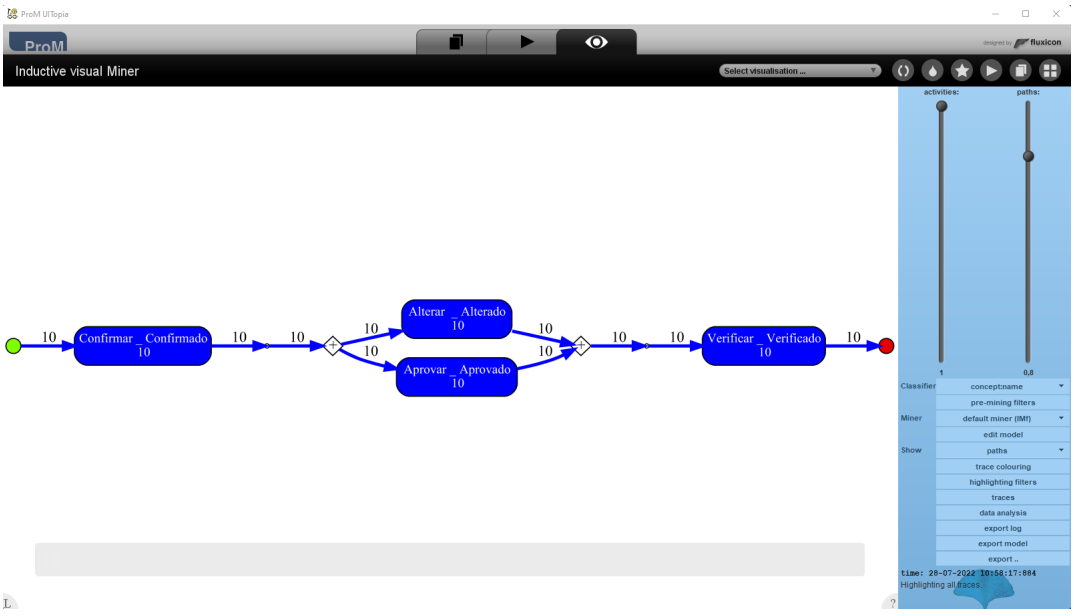

Figure 2.7: ProM Petri Net Example.

Commercial process mining tools offer little assistance for creating customized algorithms. Furthermore, both commercial and open-source process mining technologies are frequently only available via a graphical user interface, which limits their application in large-scale experimental settings [12].

Process Mining for Python (PM4Py) framework, developed by the Fraunhofer Institute for Applied Information Technology, emerged from the need to have a process mining software that was easily extensible, that provided for algorithmic customization, and that enabled the user to execute large-scale experiments with ease. There are several reasons that this library was very beneficial to the process mining discipline [12]:

- Increased the possibility of algorithmic development and customization in process mining analysis, compared to the existing open source tools.
- Process mining algorithms can be easily integrated with algorithms from other data science domains.
- It built a collaborative environment in which researchers and practitioners can share useful code and observations with the process mining community.
- It is user-friendly since it has thorough documentation.
- Intensive testing ensures algorithmic stability.

This framework was absolutely crucial in this thesis, the influence it had on the development of the desired application will be demonstrated in Chapter 3.

2.7 Synthesis of the State of the Art

In this Chapter we overview the importance of a document and management system and what an online process mining tool could influence positively in this market. It was made a literature review in the process mining discipline, and its techniques, with a focus in discovery. It is possible to summarize this method in terms of input and output as we can see in the Figure 2.8 [10].

Figure 2.8: Process mining discovery scheme in terms of input and output (Adapted from: [10]).

The input of a process discovery procedure is a event log, with a process model, in a modelling notation (such as Petri nets), as the output.

Conformance checking has the goal to compare an event log to a process model in order to identify deviations. This deviations are crucial to the enhancement technique that can analyse them with the goal to enhance the previous produced model.

Process mining is still an evolving technology, which is why there are still some methodological challenges that need to be overcome.

Finally, there exist several proposed tools in the process mining category, ProM being the most used tool in this field. However, PM4Py is gaining more popularity everyday since it emerged from the need to have a tool with custom process mining analysis, flexibility in the integration with another domains and with the ability to have a large-scale experimental setting for intensive algorithmic testing.

Chapter 3

Development

3.1 Description of the general plan

Before diving into the general plan, it is important to recall the main objectives of this thesis. The **primary goal** is the development of a tool capable of extracting processes from event logs **creating a process model**. The **secondary goal** is to make the most reliable process model that describes the intended process, by comparing the different process discovery algorithms.

In order to create the online tool several crucial steps took place, the Figure 3.1 is a flow chart representing the structure of the work plan behind the end goal:

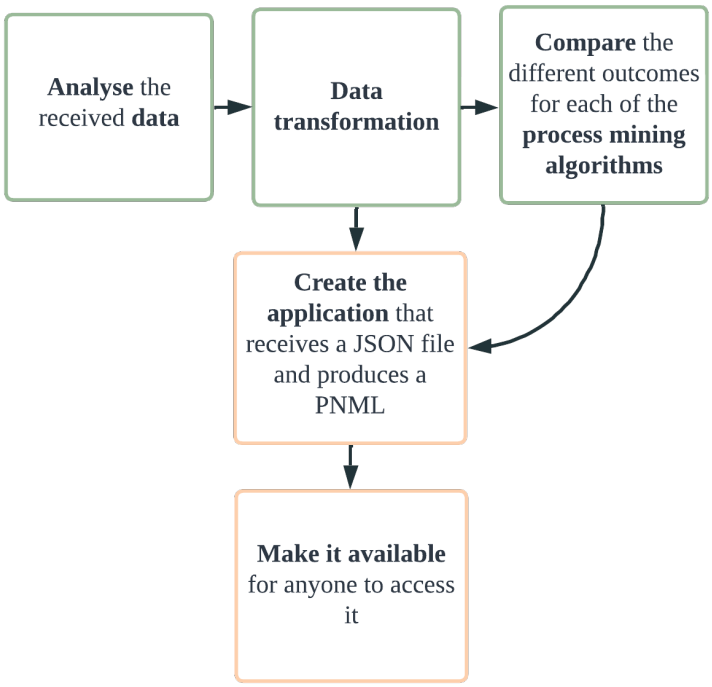

Figure 3.1: Work Plan Flow Chart.

First we decided to contact IPBrick in order to plan how the data would be retrieved. In a preliminary phase of this work, the company decided to give the necessary data by accessing it through the DBeaver (Universal database client) software for data analysis. In order to access the data, a VPN was needed and the correspondent database name, host name and password were required.

Later in the project, the company decided to give the data in a JSON format, in which, each of the file was correspondent to a process. The modeling notation chosen to retrieve the process models were Petri Nets since they are the lead process model format in discovery. The process model was sent in Petri Net Markup Language (PNML) format, which is an Extensible Markup Language (XML) based exchange format for Petri nets.

After studying the necessary data and making the required data transformations, the beginning of the development of the application and the algorithm testing begun. The JSON files serves as an input to the Process Extrator and Updater Tool. Finally, by making the application available, the company and anyone that intends to create/update their existing models will be able to do so.

In order to create the interface for the online platform, the Streamlit framework was essential. It is an open-source app framework for Machine Learning and Data Science teams that supported the visualization of graphs which was a major priority in this context.

Docker and Portainer softwares were needed since it created a container for the application in order to make it available.

All of the work was developed in Python 3.9.2 in which several packages were used and will be mentioned throughout this chapter. However, a special recognition to the Process Mining for Python (PM4Py) library, mentioned in the Chapter 2.6, that was crucial in order to manipulate and import the necessary data and to use and analyse the different process discovery algorithms.

In the next sections there will be a thorough description of how the data analysis was performed (see section 3.2), the necessary data transformation (see section 3.3) and the methodology that took place (see section 3.4).

3.2 Data Analysis

The data delivered from IPBrick was initially captured via SQL queries using the Psycopg2 package. This was very beneficial in order to better understand the data that we were going to work with and the different contents the process could entail.

However, this method wasn't the way we collected the data for the online tool. Updating the processes by reading the data through SQL queries wouldn't be beneficial to the company, since this would be time consuming, error prone and not automatic. In the Chapter 3.2.1 it will be explained how the data collection for the application was done.

The database had a total of 1063 tables, in which 8 were the ones that contained the information needed to describe a process.

The selection of the 8 tables was done with the SQL query found in the code block 3.1. The chosen tables where: *accaorev*, *estadorev*, *workflowrev*, *revisaodoc*, *documento*, *mailutilizador*, *ligadocsec* and *seccao*.

```
select  idaccaorev,
        accaorev.idestadorev, accaorev.idaccao, accaorev.idpessoa,
        accaorev.datainicio, accaorev.datarealizacao, accaorev.descricao,
        accaorev.idmultiplos, accaorev.docid, accaorev.limitdate,
        estadorev.realizado, estadorev.idworkrev, estadorev.idestado,
        workflowrev.idwork, workflowrev.idrev, workflowrev.finalizado,
        mailutilizador.utilizador, mailutilizador.mail,
        revisaodoc.codigo
from accaorev
inner join estadorev on accaorev.idestadorev = estadorev.idestadorev
inner join workflowrev on workflowrev.idworkrev = estadorev.idworkrev
inner join revisaodoc on workflowrev.idrev = revisaodoc.idrev
inner join documento on revisaodoc.iddoc = documento.iddoc and accaorev.docid =
    documento.iddoc
inner join mailutilizador on mailutilizador.utilizador = revisaodoc.autor
inner join ligadocsec on ligadocsec.iddoc = documento.iddoc
inner join seccao on seccao.idsec = ligadocsec.idsec
```

Listing 3.1: SQL query for data selection.

The Figure 3.2 shows a description of the variables found in the resulting data frame, created with the Pandas package, after the SQL query was performed.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210775 entries, 0 to 210774
Data columns (total 19 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   idaccaorev      210775 non-null  int64
 1   idestadorev     210775 non-null  int64
 2   idaccao         210775 non-null  int64
 3   realizado       210775 non-null  bool
 4   idpessoa        210775 non-null  int64
 5   datainicio      210775 non-null  datetime64[ns]
 6   datarealizacao  180798 non-null  object
 7   descricao       210775 non-null  object
 8   idmultiplos     15727 non-null   float64
 9   docid           210775 non-null  int64
 10  limitdate       210775 non-null  datetime64[ns]
 11  idworkrev       210775 non-null  int64
 12  idestado        210775 non-null  int64
 13  idwork          210775 non-null  int64
 14  idrev           210775 non-null  int64
 15  finalizado      210775 non-null  bool
 16  utilizador      210775 non-null  int64
 17  mail            210775 non-null  object
 18  codigo          210775 non-null  object
dtypes: bool(2), datetime64[ns](2), float64(1), int64(10), object(4)
memory usage: 27.7+ MB
```

Figure 3.2: Data description after SQL query.

We can conclude that, only 2 of them had null values (*datarealizacao*, *idmultiplos*), 10 were integers (*idaccaorev*, *idestadorev*, *idaccao*, *idpessoa*, *docid*, *idworkrev*, *idestado*, *idwork*, *idrev*, *utilizador*), 2 where boolean (*realizado*, *finalizado*), 2 were dates (*datainicio*, *limitdate*) and finally

4 where categorical (*datarealizacao*, *descricao*, *mail*, *codigo*).

There exist several proposed process mining technologies, as mentioned in the State of the Art (Chapter 2), and within each type, it may differ the desire of how descriptive the process is wanted, for example, in process mining enhancement it is beneficial to have additional information about the process in order to find patterns and discrepancies.

In process discovery, the goal is to make the least complex model possible that can describe the workflow of a process. With that into consideration, in discovery, usually there is no need to have additional information in a process, since the basic information (case, activity and timestamp) is often sufficient to correctly describe a process.

With that into consideration, only 4 variables where chosen to analyse the processes, the timestamp (*datainicio*), the activity (*descricao*), the case identifier (*idworkrev*) and finally the process identifier (*workid*). In the Table 3.2 it is shown the first 5 rows of the resulting data (out of 210795 entries).

	datainicio	descricao	idworkrev	idwork
168395	2016-01-20 10:31:48	Email Reply	3669	128
190185	2016-01-20 15:34:09	Email Reply	3670	128
190188	2016-01-21 15:55:17	Encaminhar Circular Informativa para assinatura	3671	195
190189	2016-01-21 15:56:32	Permissão de leitura	3671	195
862	2016-01-21 15:57:06	Divulgar Circular Informativa	3671	195

Table 3.1: Data frame of the selected variables to describe the process.

In order to have a better knowledge of the distribution of the activities and instances in a process, two graphical representations where made. First, a data frame was created that contained the number of activities and instances per process.

In order to make the desired dataframe, first it was stored the unique values of the process identifier in a list. Then in a new dataframe the process identifier, the number of activities and the number of instances were appended. To collect the number of activities the 'descricao' parameter was used and for the number of instances the 'idworkrev' parameter.

In the Table 3.2 it is shown the first 5 rows of the resulting data (out of 69 processes).

	Process	Number of Activities	Number of Instances
0	128	11	37277
1	250	15	205
2	268	17	156
3	194	2	6
4	165	20	34

Table 3.2: Data frame of activities and instances per process.

Secondly, using the Matplotlib package, the distribution of the number of instances per process was translated in the histogram shown in the Figure 3.3.

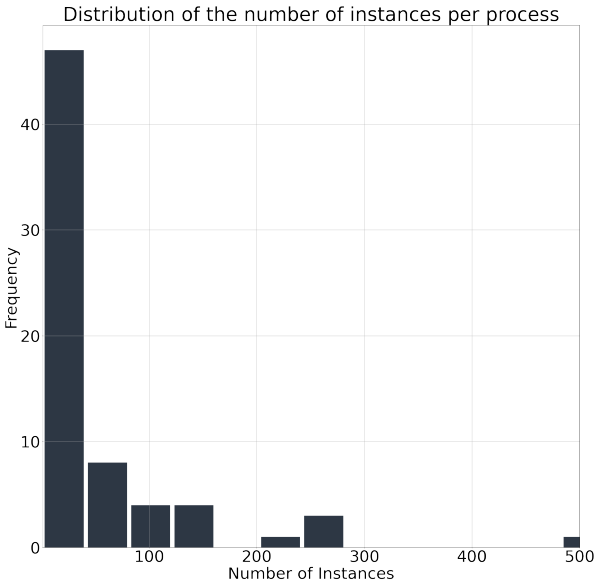

Figure 3.3: Distribution of the number of instances per process.

The process 128, that was previously presented in the Table 3.2, is not plotted in the graph, meaning that the high value of instances in this process was an outlier. We can conclude that most of the processes have each a number less or equal to 150 instances.

To represent the distribution of the number of activities per process the same procedure was applied, depicted in the Figure 3.4.

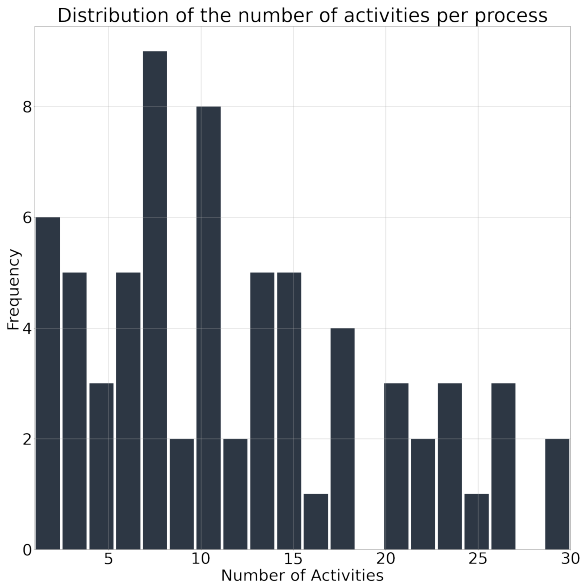

Figure 3.4: Distribution of the number of activities per process.

It is possible to conclude that although there is a good distribution of the number of activities per process, there is a slight more concentration in the number of processes with 15 or less activities each.

3.2.1 Data Collection Alternative

The data received by IPBrick and analysed via the SQL query, mentioned in the previous Section, was essential to the better understanding of what type of data we would be working with. In order to automatically update the process models effectively the created tool didn't receive the data via SQL queries.

Instead, as mentioned in the Section 3.1, IPBrick decided to send the data via JSON files that were correspondent to a process. This file would serve as the input to the online tool, which the goal was to retrieve the correspondent process model. More information about how this was performed will be in the section 3.4.

3.3 Data Transformation

The data that was received in JSON format needed to be converted to a event log compatible with the PM4Py techniques. First we normalized the JSON file to a dataframe and then decided which of the columns were the case, activity and timestamp.

In the Table 3.3, it is represented an example of a received process from IPBRick after being normalized to a dataframe using the Pandas package.

	idworkrev	descricao	datarealizacao	idmultiplostemp	resultado	idpessoa
0	36373	Assinar Consentimento Informado - Utente (Admi...	2020-07-14 13:38:32+01	4926	Assinado	758
1	36373	Integrar com outros sistemas	2020-07-14 13:38:51+01	4927	Executar	10022
2	36373	Confirmar dados do Utente	2020-07-14 13:36:15+01	4921	Confirmado	10022
3	36373	Assinar documento - Médico (SCard - CC)	2020-07-14 13:37:16+01	4923	Assinado segue p/ Administrativos	10022
4	36375	Confirmar dados do Utente	2020-07-14 14:01:14+01	4921	Confirmado	10022

Table 3.3: Normalized dataframe.

After analysing the information received, we concluded that the case identifier was correspondent to the *idworkrev*, the activity to *descricao* concatenated with *resultado* and the timestamp to *datarealizacao*.

The variable *descricao* was modified to contain the *resultado* information since this was key in order to correctly describe the workflow of the process. The columns *idmultiplostemp* and *idpessoa* were dropped since it wasn't necessary in this context. The *datarealizacao* column was converted to *datetime64* type using the PM4Py package and the resulting dataframe was sorted by the timestamp (ascendent). The PM4Py package supports CSV/standard pandas dataframes as well as XES formats.

In order to convert the resulting dataframe in a event log, it was important to map:

- the **'idworkrev'** column to case, which for PM4Py is the **'case:concept:name'** label;
- the **'descricao'** column as the activity, which for PM4Py is the **'concept:name'** and
- the **'datarealizacao'** column to timestamp, which for PM4Py is the **'time:timestamp'** label.

The following function was created to generate the desired event log:

```
def generateLog(df,logName):
    event_log = pm4py.format_dataframe(
        df,
        case_id = 'idworkrev',
        activity_key = 'descricao',
        timestamp_key = 'datarealizacao',
        timest_format = '%Y-%m-%d %H:%M:%S%z'
    )
    xes_exporter.apply(event_log, './log/'+logName+'.xes')
    log = xes_importer.apply('./log/'+logName+'.xes')
    return log
```

Listing 3.2: Python code to generate a event log.

With the event log created and saved in a XES file, it is now possible to generate different process models like Petri nets, heuristic nets, or process trees using the PM4Py framework. The Figure 3.5 presents a snippet of the transformed data in a XES file.

```
<?xml version="1.0" encoding="utf-8" ?>
<log xes.version="1849-2016" xes.features="nested-attributes" xmlns="http://www.xes-standard.org/">
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.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="35907" />
    <event>
      <string key="idworkrev" value="35907" />
      <string key="descricao" value="Confirmar dados do Utente _ Confirmado" />
      <date key="datarealizacao" value="2020-07-10T16:37:56+00:00" />
      <string key="concept:name" value="Confirmar dados do Utente _ Confirmado" />
      <date key="time:timestamp" value="2020-07-10T16:37:56+00:00" />
      <int key="@@index" value="12" />
    </event>
    <event>
      <string key="idworkrev" value="35907" />
      <string key="descricao" value="Assinar documento - Médico (SCard - CC) _ Assinado segue p/ Administrativos" />
      <date key="datarealizacao" value="2020-07-13T10:13:03+00:00" />
      <string key="concept:name" value="Assinar documento - Médico (SCard - CC) _ Assinado segue p/ Administrativos" />
      <date key="time:timestamp" value="2020-07-13T10:13:03+00:00" />
      <int key="@@index" value="13" />
    </event>
```

Figure 3.5: Snippet of the transformed data in a XES file.

Each trace represents a process instance, whereas each activity is designated as an event. As a result, a trace is a collection of events, and the log is a collection of traces.

3.4 Methodology

The methodology behind the online tool focuses in process mining discovery. In order to create the desired process model, it is important not only to chose the modelling notation of preference, but what discovery algorithms are tested in order to have a variety of results that are relevant for quality comparison.

The PM4Py library provides three discovery algorithms focused on Petri Net outputs. This package also implemented the Correlation Miner algorithm for directly-follows graph, however, since Petri Nets are the lead modeling notation in process mining, we decided to focus our study in this modeling language. With that into consideration, we decided to test the Alpha Miner, Heuristic Miner, and Inductive Miner (and its variants).

The Table in 3.4 [20], intends to summarize and recall the general aspects of each of these algorithms. It shows that Alpha Miner is the algorithm that is the least capable of resulting in a reliable process model. Since the retrieved database is prone to human error, this algorithm would most likely suffer reliability due to noise. In the other hand, Inductive Miner seems to be the one with the most capabilities to construct a well described model.

Alpha Miner	• It does not consider frequencies which makes it weak against noise • Cannot ensure soundness • Can't handle loops of length one and length two • Invisible and duplicated tasks can't be discovered
Heuristic Miner	• It considers frequencies which helps filtering out noise • Cannot ensure soundness • Capable of detecting short loops • It allows for the omission of invisible activities.
Inductive Miner	• It considers frequencies which helps filtering out noise • Can ensure soundness • Capable of detecting loops • Can handle invisible tasks • Most used process mining algorithm

Table 3.4: Process Discovery algorithms overall comparison (Adapted from: [20]).

In the application there will be the output of each algorithm, and a study about the quality dimensions will be done in order to correctly compare them. Before jumping into the results, its important to grasp all the work that was done in order to create the online app, which will be described in the Section 3.4.1.

3.4.1 Streamlit Application

Streamlit was founded in 2018 to assist data scientists in the development of data apps, with a focus on rapid development using a pure Python approach [31].

Streamlit is a data app framework that is also known as a data dashboarding tool. Data applications are an excellent tool for data scientists to showcase their results. Whenever the app's state changes, Streamlit re-runs the whole Python script. This allows for a high level of interaction without the use of specific callback functions [31].

Streamlit was the chosen framework in order to create this application. The framework is user-friendly with its detailed documentation and it is compatible with the majority of Python libraries (e.g. pandas, matplotlib, seaborn, plotly).

The first thing that was created in this application was the method of how the information would enter the system. It was decided that an uploading system would be an effective way to do so. This was an iterative strategy that allows not only IPBrick to use the application, but anyone with a JSON file (that refers to a process). In the Figure 3.6 how this button looks in the application.

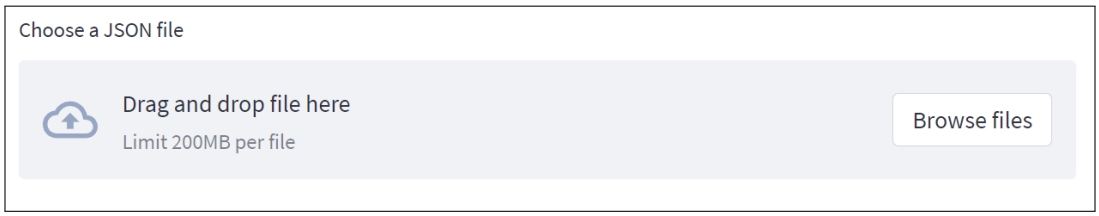

Figure 3.6: Upload Button in the App.

When the JSON files enters the system, the data transformation is performed automatically. All the data transformation steps, described in Section 3.2, are done internally. As we can see in the Figure 3.7, after uploading a JSON file, it is shown not only the content of the JSON that was uploaded but the resulting dataframe that serves as the event log.

An animation detail, in form of a progression bar, was added in order to give the user a more interactive and visual experience.

In the application we also built a dropdown widget with the different process discovery algorithms, shown in the Figure 3.8. When a algorithm is selected it shows a graphic representation of the resulting model, in the section 4 we will show how it looks in the application.

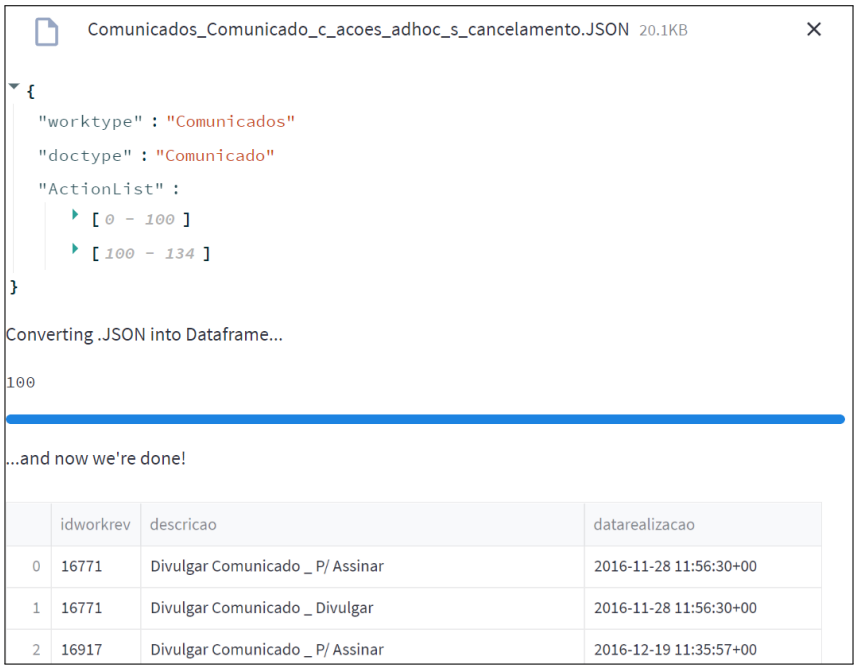

Figure 3.7: Upload Button Output.

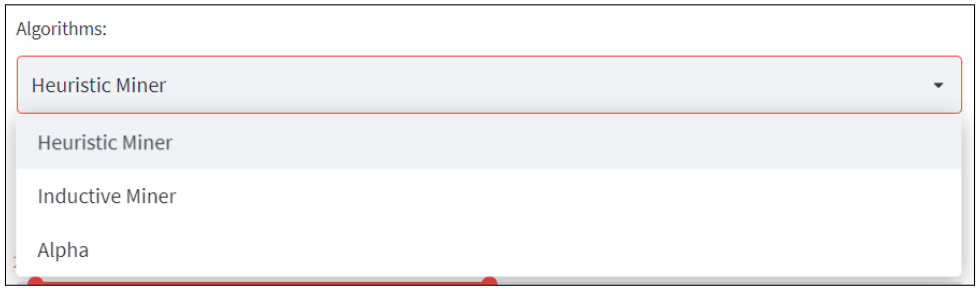

Figure 3.8: Dropdown Widget.

In this project we have a bigger focus in Petri Nets, however, for comparison purposes, it is available a representation of the correspondent Process Tree in the Inductive Miner algorithm and a Heuristic net in the case of the Heuristic Miner algorithm.

PM4Py provides an implementation of the Inductive Miner (IM), Inductive Miner Infrequent (IMf), and Inductive Miner Directly-Follows (IMd) algorithms [20]. The variant IM produces a model with perfect replay, IMf produces a more precise model, without fitness guarantees, by eliminating some behavior and finally, IMd considers directly-follows graph for maximum performance, however, it loses the ability to guarantee replay fitness.

Taking the different variations the Inductive Miner algorithm, provided by the PM4Py framework, into account, we decided to integrate the possibility to switch variants in a check box widget as can be seen in the Figure 3.9 .

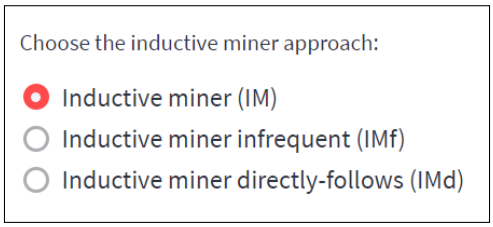

Figure 3.9: Check Box for the Inductive Miner variants.

The PM4Py framework also supplies the ability to make several filters to the data, in this work we decided to make use of the case size filter. The case size filter maintains only cases in the log that have a number of events that fall inside a user-specified range. This can serve two purposes: removing cases that are either too short (and hence clearly incomplete or outliers) or too long (therefore needing extensive rework) [20].

In order to give the user the freedom to chose the desired filter, it was added to the application, a range slider widget that was preset with a minimum of 0 and a maximum of 150 cases, Figure 3.10.

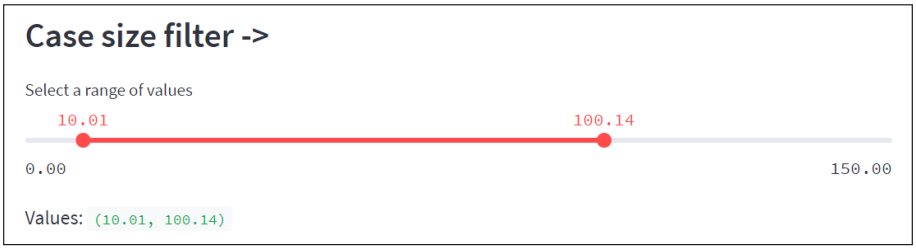

Figure 3.10: Slider for the case filter.

In the application it is also possible to extract the PNML of the respective models and the analysis of the resulting models will also be available for study. This demonstrations will be shown in the Chapter 4.

Finally, after the application was developed, it was important that it had accessibility to anyone and everywhere. That's when the Docker and Portainer software were absolutely essential.

Docker is a tool that provides rapid development, testing, and deployment of programs. Docker organizes software into standardized units called containers, which include everything the software requires to execute, such as libraries, system tools, code, and runtime. Docker allows the user to swiftly deploy and grow apps into any environment while remaining certain that their code will execute [35].

Portainer's multi-cluster, multi-cloud container management tool that supports several platforms such as Docker and Kubernetes in any data center, cloud, network edge, or device. This software facilitates and improves the deployment and management of containerized applications and services [34].

This part of the project was done in Linux-Ubuntu where it was made easier to install the necessary python packages for Docker images. First a DockerFile was created with the mandatory installations and then a requirements text file, which is supposed to contain package names and versions required by the model app, containing the streamlit and graphviz packages. In the Figure 3.11 it shows the content of the dockerfile.

```
FROM python:3.9.2
EXPOSE 8501
WORKDIR /app
COPY requirements.txt ./requirements.txt
RUN pip install -r requirements.txt
RUN pip install --upgrade pip
RUN pip install pm4py
RUN pip install streamlit
RUN pip install Pillow
RUN pip install pydot
RUN pip install pydotplus
RUN pip install graphviz
RUN pip install DateTime
RUN python -m pip install --upgrade pip
RUN pip3 install pandas
RUN pip3 install -U scikit-learn
RUN apt-get update && apt-get install -y \
  apt-utils \
  strace \
  graphviz \
  && rm -rf /var/lib/apt/lists/*
COPY . .
CMD streamlit run app.py
```

Figure 3.11: Dockerfile.

In the Figure 3.11 it is possible to grasp the several commands a Dockerfile possesses, in summary:

- **FROM**: docker is instructed to develop a foundation layer. In this scenario, we utilized the Python v3.9.2 image from Docker Hub as the foundation image for our application.
- **EXPOSE**: defines the network port number in which the application would be launched. The Streamlit library is configured to use Port 8501, which is our case.
- **WORKDIR**: defines the working directory. In this scenario, all the files will be located in the '/app' folder.
- **COPY**: replicates the local requirements file into the '/app' folder.
- **RUN**: defines which command should be performed within the Docker container (in the command line interface). In this scenario, Dockerfile installs all of the packages written in the requirements file, as well as runs some additional package downloads.
- **CMD**: is used to define the instructions that will be executed when the Docker container is launched. In this scenario, the command to run the application.

In order to create the Docker container three commands where needed:

1. **docker login**, login to the Dockerhub account,
2. **docker build -f Dockerfile -t app:latest .**, build the Docker image, -f refers to the name of the Dockerfile and -t to the tag of the image, in this case 'latest', and
3. **docker run -p 8501:8501 app:latest**, run the application from the docker container, the -p parameter is used to map the port of the container to the host port. In this case we use the port 8501.

In Portainer, IPBrick created their own account and provided for the credentials. To store the application in this platform we needed its image. This was done by creating a repository for the application in Docker Hub. In the Figure 3.12 it is shown how the repository looks in this platform.

To create the repository in Docker Hub, the following line commands where required [17]:

1. **docker build -t 'hub-user'/'repo-name'[:'tag']**, to label the local image using the Docker Hub login.
2. **docker tag 'existing-image' 'hub-user'/'repo-name'[:'tag']**, label the repository name that was generated with Docker Hub.
3. **docker commit 'existing-container' 'hub-user'/'repo-name'[:'tag']**, to commit changes.
4. **docker push 'hub-user'/'repo-name':'tag'**, push the repository to the registry specified by its name and tag.

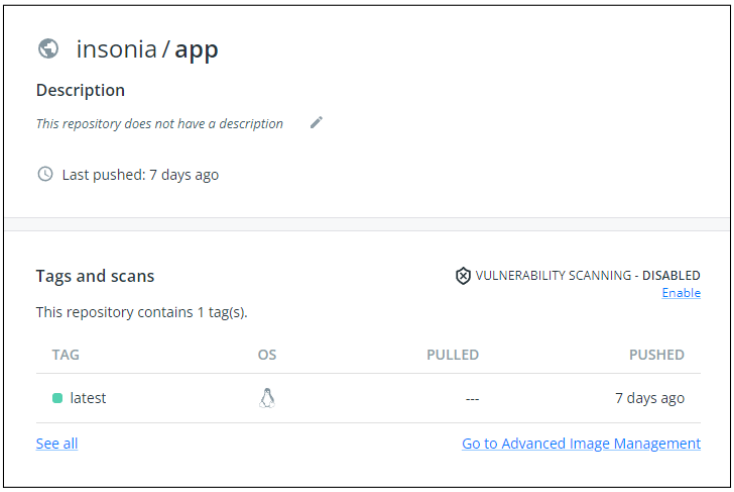

Figure 3.12: Repository of the application in DockerHub.

In portainer, the container for the application was created by accessing the image through DockerHub. After all these steps, the application was up and running, Figure 3.13 shows how the container looks in Portainer.

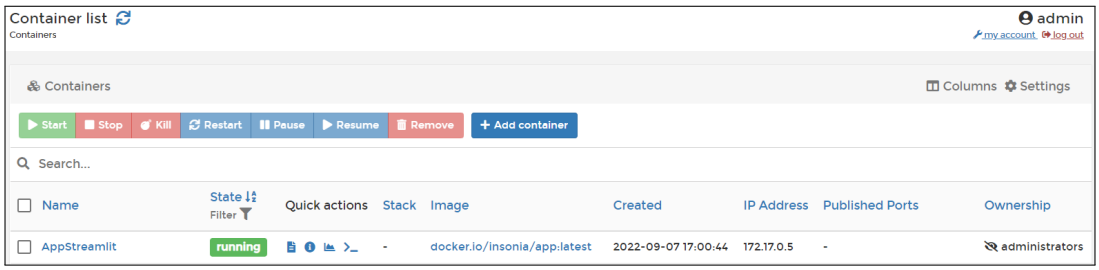

Figure 3.13: Container of the application in Portainer.

The application was then running on the url http://195.23.114.203:8501/, that only required for the port to be accepted in the firewall.

Chapter 4

Results and analysis

4.1 Introduction

This chapter is composed by two parts, the **Cases of Study and Demonstration** in the Section 4.2 and the **Results and Analysis** in the Section 4.3. In this Chapter we will overview the overall testing of the created application and the evaluation of the retrieved models.

It was important to have different processes in terms of complexity and noise for a diverse demonstration, hence why it was decided to chose two processes. In the testing section another process is going to be used since it had a more balanced noise and medium complexity in order to give a fair chance to the weaker algorithms.

In order to have a variety of processes to chose from, IPBrick shared several JSON files that was tested in the application which was very helpful for debugging purposes.

The four quality dimensions as well as soundness will be the quality criterion's for our Petri Nets.

4.2 Cases of Study and Demonstration

Uppon receiving several JSON files from IPBrick, and after testing the different possibilities, it was decided to demonstrate two very different processes in terms of complexity and noise.

4.2.1 First process demonstration

Starting with the least complex process, called 'Correspondência'. This is the process of archiving and forwarding correspondences as well as the acceptance of it. This file was sent excluding adhoc and cancelled actions in order to give us the true workflow of the process. Before jumping into the demonstrations, we decided to retrieve the model that iPortalDoc had for this kind of

process as can be seen in the Figure 4.1.

iPortalDoc uses a state machine to create their workflows, they are based on states, transitions and actions. The states are correspondent to the circles, the transitions to the strokes and the actions are in the little boxes above each state.

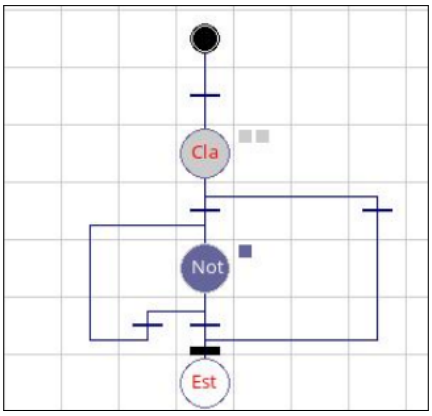

Figure 4.1: iPortalDoc model for 'Correspondência' process.

First we uploaded the JSON file into our application, and then the dataframe was created based on this file. In order to have a visual idea of the structure these outputs are shown in the application, the JSON file in the Figure 4.2 was modified, as described in the Chapter 3 Section 3.3, and resulted in the dataframe in Table 4.1.


```
{
  "worktype" : "Correspondência"
  "doctype" : "Correspondência"
  "ActionList" :
    [
      0 : {
        "idworkrev" : "16455"
        "descricao" : "Classificar e encaminhar correspondência"
        "datarealizacao" : "2016-09-26 22:56:27+01"
        "idmultiplostemp" : "4088"
        "resultado" : "Encaminhar"
        "idpessoa" : "10167"
      }
      1 : {
        "idworkrev" : "16618"
        "descricao" : "Classificar e encaminhar correspondência"
        "datarealizacao" : "2016-11-03 11:39:18+00"
        "idmultiplostemp" : "4088"
        "resultado" : "Encaminhar"
        "idpessoa" : "10233"
      }
      2 : {
        "idworkrev" : "16637"
        "descricao" : "Classificar e encaminhar correspondência"
        "datarealizacao" : "2016-11-03 16:43:32+00"
        "idmultiplostemp" : "4088"
        "resultado" : "Encaminhar"
        "idpessoa" : "10233"
      }
```


Figure 4.2: 'Correspondência' process JSON file.

	idworkrev	descricao	datarealizacao
28	6727	Classificar e encaminhar correspondência _ Enc...	2016-05-02 16:34:29+01
32	6805	Classificar e encaminhar correspondência _ Enc...	2016-05-04 12:56:39+01
33	6805	Tomar conhecimento de correspondência _ Arquivar	2016-05-04 13:04:08+01
29	8790	Classificar e encaminhar correspondência _ Enc...	2016-05-17 17:55:49+01
30	8790	Tomar conhecimento de correspondência _ Arquivar	2016-05-17 17:59:28+01
34	10345	Classificar e encaminhar correspondência _ Enc...	2016-05-31 17:15:56+01
31	11189	Classificar e encaminhar correspondência _ Enc...	2016-06-07 17:11:25+01
35	11604	Classificar e encaminhar correspondência _ Arq...	2016-06-16 15:00:18+01
76	16350	Classificar e encaminhar correspondência _ Enc...	2016-08-25 18:44:42+01
77	16350	Tomar conhecimento de correspondência _ Arquivar	2016-08-26 11:39:38+01

Table 4.1: 'Correspondência' process dataframe.

Now that we have the necessary data to apply process discovery, lets analyse the resulting Petri Net and Heuristic Net from the Heuristic Miner algorithm. In the Figure 4.3 and its the output of the Heuristic Net and in the Figure 4.4 the output of the Petri Net.

As we can observe, the Heuristic Net and the Petri Net showed that there exist some outliers, for example the self loop in 'Tomar conhecimento de correspondência_Encaminhar', the 'Classificar e encaminhar correspondência _Encaminhar' has an arc that goes directly to the end of the cycle and an arc that goes directly to 'Tomar conhecimento de correspondência_Arquivar'.

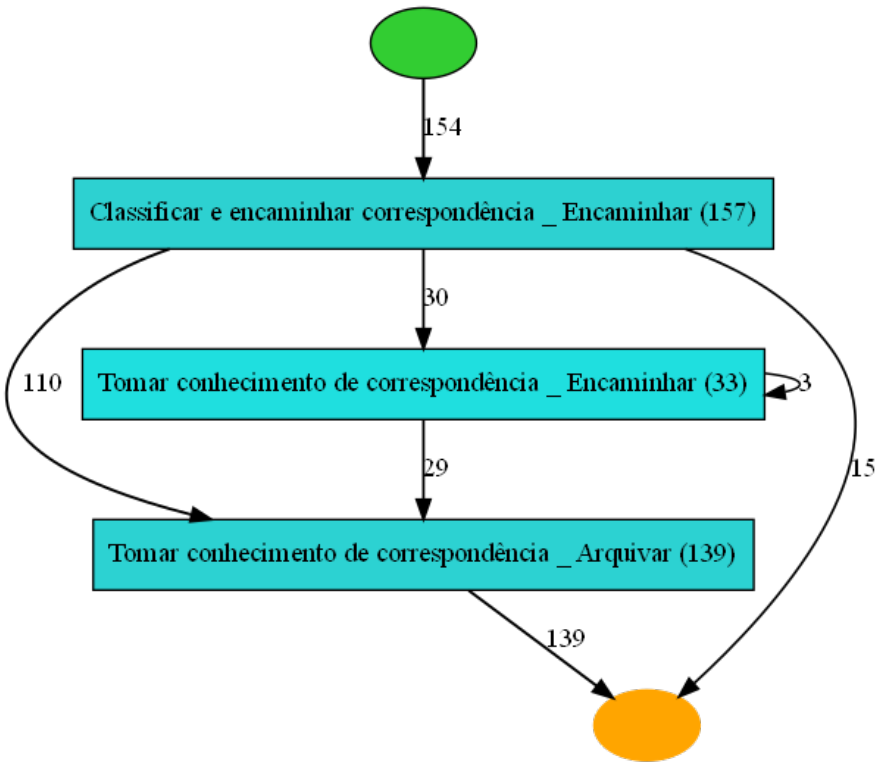

Figure 4.3: Heuristic Net 'Correspondência'.

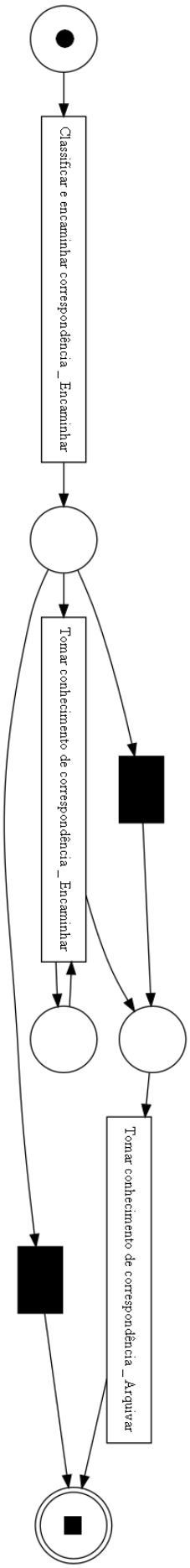

Figure 4.4: Petri Net 'Correspondência' Heuristic Miner.

Process mining presents conformance checking and enhancement techniques that helps in dealing with bottlenecks and outliers like the ones referred, however, we decided to use the Case Size filter that was added to the application to see if we could eliminate some of this behaviour. The filter was set for the cases with a minimum of 4 events and a maximum of 100 the resulting Petri net is shown in the Figure 4.6.

Since we only applied a filter it is only natural that not all of the bottlenecks were dealt with, however we see some improvement, the 'Classificar e encaminhar correspondência_Encaminhar' presents no longer an arc that goes directly to the end of the cycle, however, it created a self loop in the first transition.

As we can conclude, filtering is not ideal when it comes to dealing with discrepancies specially from data that comes from manual created processes, since it presents a higher error probability and therefore, stronger techniques are ideal in this cases.

In Process Mining for Python (PM4Py) framework Heuristic Net also comes with a parameter that helps filtering out noise and find common constructs (dependency between two activities), called "dependency treshold", we decided to implement it in the application through a range slider widget, as can be seen in the Figure 4.5.

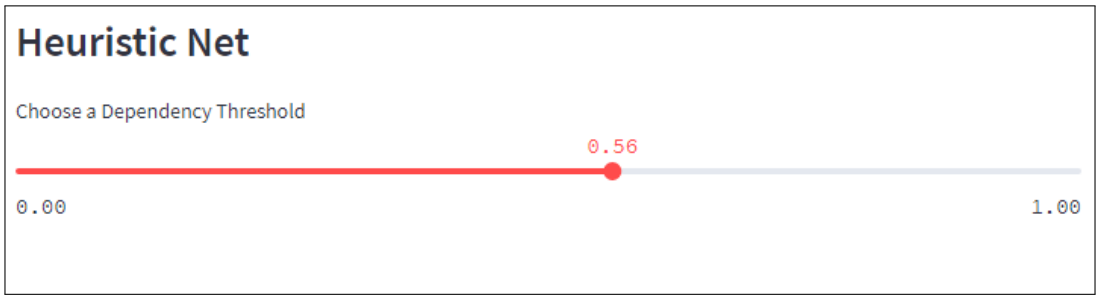

Figure 4.5: Dependency Threshold Widget.

This parameter is by default set to 0 , and has a maximum of 1, after testing out different values, the 0.57 value was the one that showed the best results and the resulting Petri Net is in the Figure 4.6 .

As we can conclude, the dependency threshold filter worked really well, it took a lot of unnecessary steps, even though there might still be some discrepancies remaining. For the other two algorithm's, we will demonstrate the output for Inductive Miner (IM) and Alpha Miner, and in the Section 4.3 we will see how all of the algorithms and variants performed.

In the IM algorithm provided by the PM4Py framework we decided to use a parameter that provides the possibility to decorate the Petri Net. The empty transitions and the places will labeled which helps in readability. The output of the IM algorithm for this process in present in the Figure 4.7 (the case size filter was maintained).

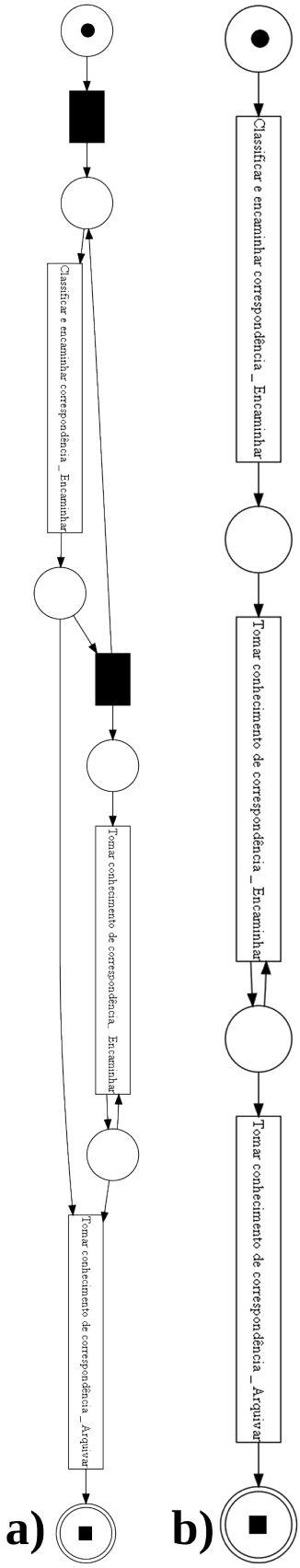

Figure 4.6: Petri Nets 'Correspondência' after a) case size filter and b) dependency threshold filter.

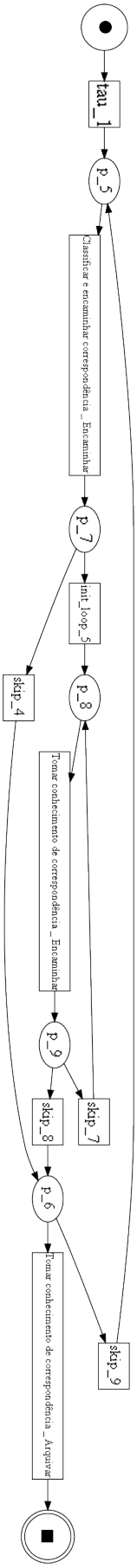

Figure 4.7: Petri Net 'Correspondência' Inductive Miner.

The IM algorithm displayed a more visually complex and complete model. In order to have other representations of this result and to better understand the process, we decided to create a Process Tree from this Petri Net, in the Figure 4.8.

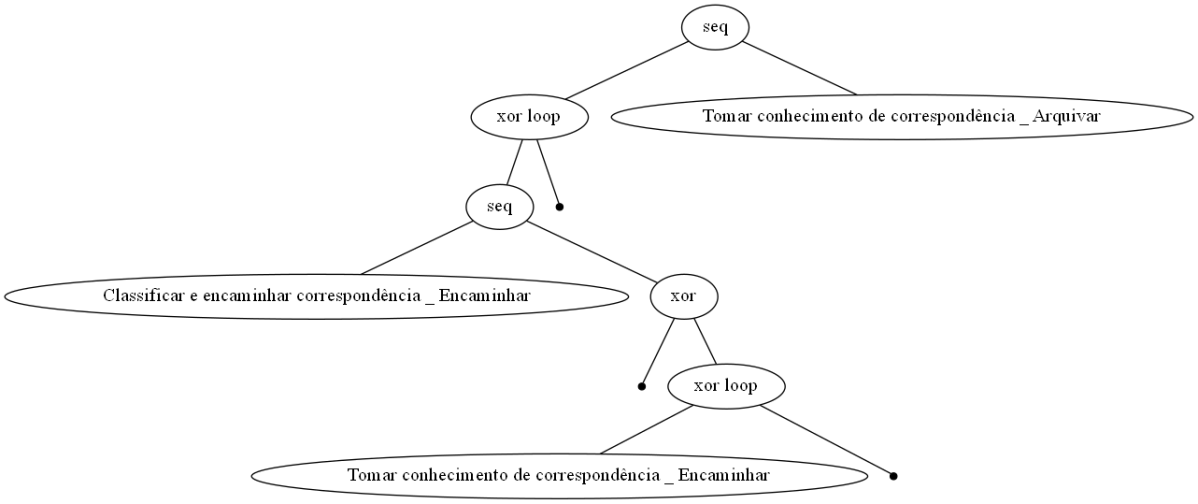

Figure 4.8: Process Tree 'Correspondência' Inductive Miner.

Finally, for Alpha Miner, the model had a really poor outcome, since this data is created manually it has some noise, and this algorithm is very weak against it, these results where expected as can be seen in the Figure 4.9 . It couldn't map the arcs and the beginning of the cycle.

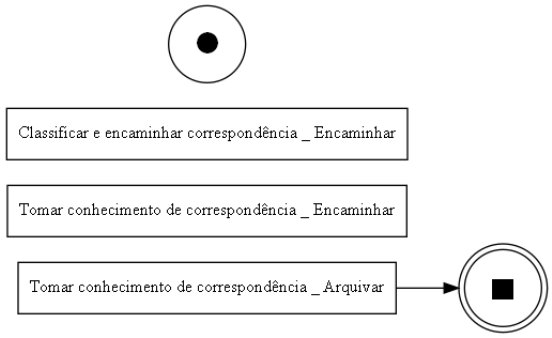

Figure 4.9: Petri Net 'Correspondência' Alpha Miner.

4.2.2 Second process demonstration

For the most complex JSON we encountered let's observe the different outcomes it displayed. This process is called 'Consentimento informado'. This process is used for medical processes, for example, for permission when someone needs some medical aftercare. In the Figure 4.10, we see the model retrieved by the iPortalDoc and comparing with the 'Correspondência' output we immediately grasp the higher complexity this process contains.

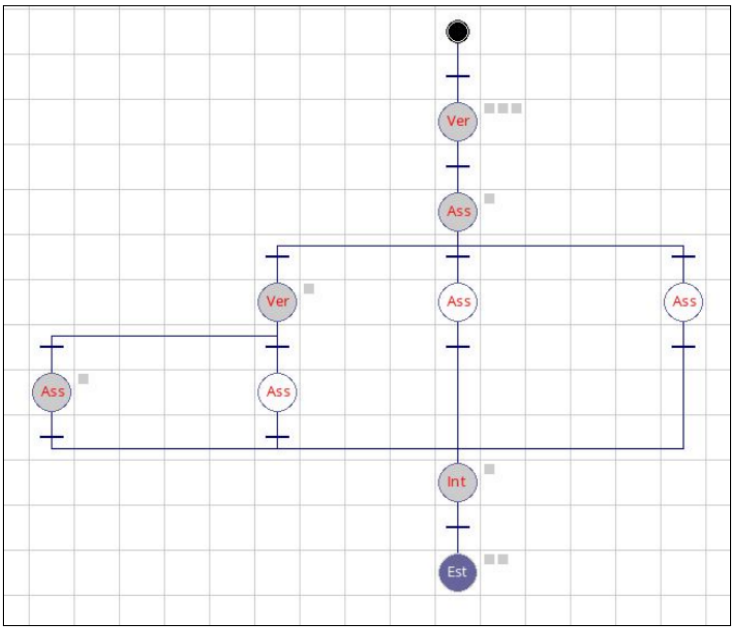

Figure 4.10: iPortalDoc model for 'Consentimento informado' process.

After uploading the JSON file into our application, the dataframe created is demonstrated in the Table 4.2.

	idworkrev	descricao	datarealizacao
12	35907	Confirmar dados do Utente _ Confirmado	2020-07-10 17:37:56+01
13	35907	Assinar documento - Médico (SCard - CC) _ Assi...	2020-07-13 11:13:03+01
2	36373	Confirmar dados do Utente _ Confirmado	2020-07-14 13:36:15+01
3	36373	Assinar documento - Médico (SCard - CC) _ Assi...	2020-07-14 13:37:16+01
0	36373	Assinar Consentimento Informado - Utente (Admi...	2020-07-14 13:38:32+01
1	36373	Integrar com outros sistemas _ Executar	2020-07-14 13:38:51+01
4	36375	Confirmar dados do Utente _ Confirmado	2020-07-14 14:01:14+01
5	36375	Assinar documento - Médico (SCard - CC) _ Assi...	2020-07-14 15:00:04+01
6	36375	Assinar Consentimento Informado - Utente (Admi...	2020-07-14 15:01:18+01
7	36375	Integrar com outros sistemas _ Executar	2020-07-14 15:01:24+01

Table 4.2: 'Consentimento informado' process dataframe.

Having the needed data to start discovery let's observe the results created by the different process mining algorithms. The algorithms retrieved a model with good readability, and after testing out different values in the case size and dependency threshold filters it was found that no improvements where being accomplished, so no filters where applied.

The models displayed a similar output for all of the algorithms, as can be seen in the Figure 4.11 where is it displayed the three outputs. We can conclude that this process, even though it had an initially bigger structure , presented little to no noise which made all of the algorithms perform in a similar way and result in a more readable model.

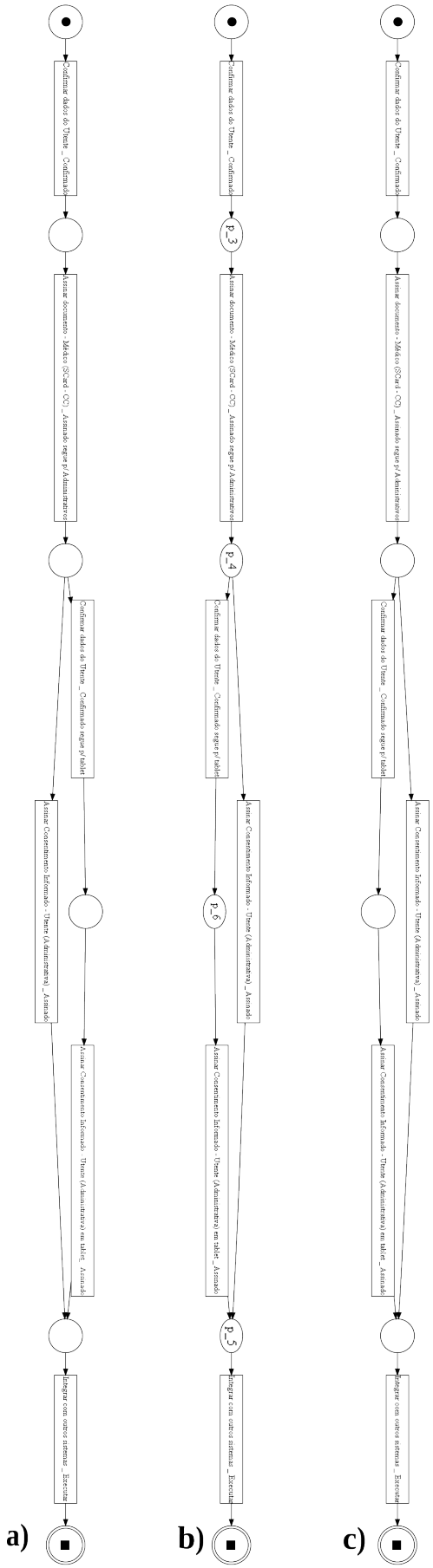

Figure 4.11: Petri nets 'Consentimento informado' a) Heuristic Miner, b) Inductive Miner and c) Alpha Miner.

In the next section, evaluation metrics will be performed to understand how to correctly choose the best algorithm given a specific process. It will also be demonstrated the content of the Petri Net Markup Language (PNML) file.

4.3 Results and Analysis

The analysis of the models where done using the four quality dimensions, described in Chapter 2 Section 2.4.1.5, and soundness, describes in Chapter 2 Section 2.4.1.3 .

For the results we decided to use a process that presents low noise in order to give a chance to algorithms like Alpha miner that struggle with this type of discrepancies and also with a structure that wasn't too complex.

The process we are going to test is 'Marcação de Férias'. This process deals with the acceptance and decline of holiday bookings. In the Figure 4.12, it is the structure of the workflow retrieved by the iPortalDoc platform.

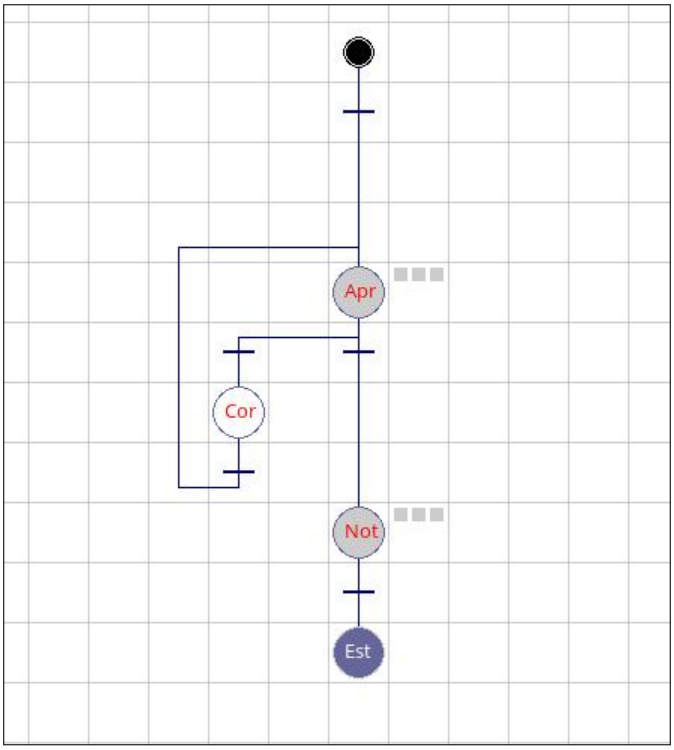

Figure 4.12: iPortalDoc model for 'Marcação de Férias' process.

After running the several discovery algorithms, it was decided to create the Table 4.3 that contained the information about the soundness of the model, replay fitness, precision, simplicity and generalization. This table allowed us to easily compare this methods. Along with the already familiar algorithms, we will include the Inductive Miner Infrequent (IMf) and Inductive Miner Directly-Follows (IMd) variants results to this analysis.

Heuristic Miner	• **Soundness:** True • **Replay Fitness:** 0.998 • **Precision:** 1.0 • **Simplicity:** 1.0 • **Generalization:** 0.842
IM variant	• **Soundness:** True • **Replay Fitness:** 1.0 • **Precision:** 0.768 • **Simplicity:** 0.777 • **Generalization:** 0.753
IMf variant	• **Soundness:** True • **Replay Fitness:** 1.0 • **Precision:** 0.768 • **Simplicity:** 0.777 • **Generalization:** 0.753
IMd variant	• **Soundness:** True • **Replay Fitness:** 1.0 • **Precision:** 0.624 • **Simplicity:** 0.777 • **Generalization:** 0.870
Alpha Miner	• **Soundness:** False • **Replay Fitness:** 0.779 • **Precision:** 0.634 • **Simplicity:** 1.0 • **Generalization:** 0.826

Table 4.3: Process Discovery algorithms results.

When analysing the several results, we can automatically see that Alpha Miner performed the worse, the model was not Sound which means that it fail in one or more of these characteristics: safeness, proper completion, option to complete and absence of dead parts.

The other two algorithms and variants had very good results, however, the best one is very apparent. Heuristic Miner is the best algorithm for this process, overall it obtained better

qualifications maintaining all of the metrics above 0.8, this has to show that even though all of the Inductive Miner variants can deal with more discrepancies when compared with Heuristic Miner, doesn't mean that it will perform better since they have different ways of constructing the model. That is why it is so important to test several algorithms depending on the chosen process.

The application provides the ability to download a PNML file for any of the algorithms. In the Figure 4.13 is it shown the resulted Petri Net for the Heuristic Miner algorithm and in the Figure 4.14 a snippet of the contents of this model in the PNML file.

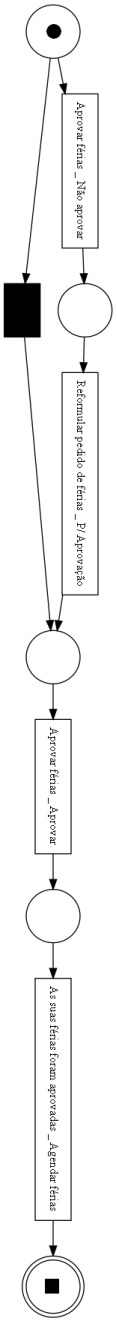

Figure 4.13: Heuristic Miner 'Marcação de Férias' Petri Net.

```
<place id="pre_Reformular pedido de férias _ P/ Aprovação">
  <name>
    <text>pre_Reformular pedido de férias _ P/ Aprovação</text>
  </name>
</place>
<transition id="Aprovar férias _ Não aprovar">
  <name>
    <text>Aprovar férias _ Não aprovar</text>
  </name>
</transition>
<transition id="Aprovar férias _ Aprovar">
  <name>
    <text>Aprovar férias _ Aprovar</text>
  </name>
</transition>
<transition id="Reformular pedido de férias _ P/ Aprovação">
  <name>
    <text>Reformular pedido de férias _ P/ Aprovação</text>
  </name>
</transition>
<transition id="As suas férias foram aprovadas _ Agendar férias">
  <name>
    <text>As suas férias foram aprovadas _ Agendar férias</text>
  </name>
</transition>
<transition id="hid_2">
  <name>
    <text>hid_2</text>
  </name>
  <toolspecific tool="ProM" version="6.4" activity="$invisible$" localNodeID="c5fcc2ba-de96-41c1-ad48-80a283aa71f4"/>
</transition>
<arc id="1656033070432" source="source0" target="hid_2"/>
<arc id="1656033070624" source="Reformular pedido de férias _ P/ Aprovação" target="pre_Aprovar férias _ Aprovar"/>
<arc id="1656033070720" source="pre_Reformular pedido de férias _ P/ Aprovação" target="Reformular pedido de férias _ P/ Aprovação"/>
<arc id="1656033069664" source="Aprovar férias _ Aprovar" target="pre_As suas férias foram aprovadas _ Agendar férias"/>
<arc id="1656033070048" source="As suas férias foram aprovadas _ Agendar férias" target="sink0"/>
<arc id="1656033069856" source="Aprovar férias _ Não aprovar" target="pre_Reformular pedido de férias _ P/ Aprovação"/>
<arc id="1656033067792" source="pre_Aprovar férias _ Aprovar" target="Aprovar férias _ Aprovar"/>
<arc id="1656033070480" source="hid_2" target="pre_Aprovar férias _ Aprovar"/>
<arc id="1656033067888" source="pre_As suas férias foram aprovadas _ Agendar férias" target="As suas férias foram aprovadas _ Agendar férias"/>
<arc id="1656033070384" source="source0" target="Aprovar férias _ Não aprovar"/>
```

Figure 4.14: 'Marcação de Férias' Petri Net PNML.

Chapter 5

Conclusion

It is very apparent that process mining is still a growing technology with lots of room for improvement. However, this discipline as already proved to be essential when it comes to keeping up an organized Process Documentation System since it helps maintaining up to date data, removing errors that happen in manual settings, the avoidance of information loss, less time consumption, and the support it gives a company when it comes to controlling every process that happens in the environment.

The objective of this dissertation consisted in the development of a tool capable of extracting and updating models by receiving event logs as the input. To start this project we needed to gain a deep understanding of the process mining domain and the impact of the discovery technique.

When contacting with IPBrick and receiving the necessary data to start creating the application, we encountered two very essential frameworks for this thesis, Process Mining for Python (PM4Py) and Streamlit. The thorough documentation of these libraries were crucial in the development of this tool.

Finally, when the tool was created and ready for testing, we grasped an even better understanding of how the models looked and how the algorithms performed depending on the data that was submitted. The four quality dimensions and the soundness of a model were our baseline criteria when it came to chose the best algorithm for the specified process. The possibility to extract the Petri Net Markup Language (PNML) was also very beneficial for IPBrick that was interested in receiving Petri Nets to their database.

After testing the different process discovery algorithms we understood that Alpha Miner was in fact very weak against noise and struggled in giving accurate models, Heuristic Miner not only gave well constructed models but demonstrated possibilities to filter out some noise using the dependency threshold and finally Inductive Miner and its variants all performed with very good quality results.

All of the algorithms can perform in good standards depending on the process it is analysing, with that into consideration, intending to chose an algorithm for a specific process, it's best to

first analyse the results they give in the quality dimensions and soundness and then chose the best outcome. These results are specified in the application to be easy for anyone using it to rapidly grasp which model is best suited for their situation.

In general, all of the pre-established objectives where concluded with success, since we created an online tool capable of receiving event logs and retrieving process models.

5.1 Future Work

In future work we believe it would be very promising if the event logs were received in streams since we could then send to the company streaming data with several PNML directly to their database. This would save a lot of time in when it comes to extract and update data since a lot of files would be running at once.

This streaming scenario was one of our initial goals when it came to the application, however, the lack of information about this type of streaming service was very limitating when it came to developing this system.

Finally, in the process mining domain, it also would be very appealing to experiment conformance checking in order to better understand the discrepancies that happen in the processes and how they can be dealt with.

The received data had a lot of noise which is normal when the data comes from real scenarios due to the human errors that when data registers are generated in their own management systems, in this project the noise was a hurdle since it slowed the process of interpreting the models and the algorithms abilities, this became an important investigation field.

Bibliography

[1] Wil Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8:21–66, 02 1998. doi:10.1142/S0218126698000043.

[2] Wil Aalst. Process discovery: Capturing the invisible. *Computational Intelligence Magazine, IEEE*, 5:28 – 41, 03 2010. doi:10.1109/MCI.2009.935307.

[3] Wil Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, volume 136. 01 2011. ISBN: 978-3-642-19344-6. doi:10.1007/978-3-642-19345-3.

[4] Wil Aalst. Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems*, 3:7.1–7.17, 07 2012. doi:10.1145/2229156.2229157.

[5] Wil Aalst. *Decomposing Petri Nets for Process Mining: A Generic Approach*, volume 31. 01 2012. doi:10.1007/s10619-013-7127-5.

[6] Wil Aalst. *Process Mining: Data Science in Action*. 01 2016. ISBN: 9783662498507. doi:10.1007/978-3-662-49851-4.

[7] Wil Aalst. Process discovery from event data: Relating models and logs through abstractions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8:e1244, 02 2018. doi:10.1002/widm.1244.

[8] Wil Aalst and Boudewijn Dongen. Discovering petri nets from event logs. 01 2013. doi:10.1007/978-3-642-38143-0_10.

[9] Wil Aalst and A. Weijters. Process mining: A research agenda. *Computers in Industry*, 53: 231–244, 06 2004. doi:10.1016/j.compind.2003.10.001.

[10] Wil Aalst, Arya Adriansyah, Ana Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose R.P., Peter Brand, Ronald Brandtjen, Joos Buijs, Andrea Burattin, Josep Carmona, Malú Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, and Moe Wynn. Process mining manifesto. *Lecture Notes in Business Information Processing*, 99:169–194, 08 2011. doi:10.1007/978-3-642-28108-2_19.

[11] Alessandro Berti and Wil Aalst. A novel token-based replay technique to speed up conformance checking and process enhancement. 07 2020.

[12] Alessandro Berti, Sebastiaan van Zelst, and Wil Aalst. Process mining for python (pm4py): Bridging the gap between process- and data science. 05 2019.

[13] Alejandro Bogarín, Rebeca Cerezo, and Cristóbal Romero. Discovering learning processes using inductive miner: A case study with learning management systems (lmss). *Psicothema*, 30:322–329, 08 2018. doi:10.7334/psicothema2018.116.

[14] J. Buijs, B. Dongen, and Wil Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, 23, 03 2014. doi:10.1142/S0218843014400012.

[15] J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. A genetic algorithm for discovering process trees. pages 1–8, 2012. doi:10.1109/CEC.2012.6256458.

[16] Joos Buijs, Boudewijn Dongen, and Wil Aalst. On the role of fitness, precision, generalization and simplicity in process discovery. 7565:305–322, 09 2012. doi:10.1007/978-3-642-33606-5_19.

[17] Docker. Repositories, 2022. Accessed: 2022-09-10.

[18] Sebastian Dunzer, Matthias Stierle, Martin Matzner, and Stephan Baier. Conformance checking: a state-of-the-art literature review. pages 1–10, 06 2019. doi:10.1145/3329007.3329014.

[19] Fluxicon. Process mining book - data requirements, 2020. Accessed: 2022-03-11.

[20] Fraunhofer Institute for Applied Information Technology. State-of-the-art-process mining in python, 2021. Accessed: 2022-09-01.

[21] Francine Freddo, Sandro Sawicki, Rafael Z. Frantz, and Fabricia Roos-Frantz. Using timed and coloured petri nets for modelling, simulation, and analysis of integration solutions. *International Journal of Web Engineering and Technology*, 14:231, 01 2019. doi:10.1504/IJWET.2019.105587.

[22] Cleiton Garcia, Alex Meincheim, Elio Junior, Marcelo Dallagassa, Denise Vecino Sato, Deborah Carvalho, Eduardo Santos, and Edson Scalabrin. Process mining techniques and applications - a systematic mapping study. *Expert Systems with Applications*, 133, 05 2019. doi:10.1016/j.eswa.2019.05.003.

[23] Raji Ghawi. Process discovery using inductive miner and decomposition. 10 2016.

[24] Stijn Goedertier, David Martens, Jan Vanthienen, and Bart Baesens. Robust process discovery with artificial negative events. *Katholieke Universiteit Leuven, Open Access publications from Katholieke Universiteit Leuven*, 10, 06 2009. doi:10.1145/1577069.1577113.

[25] Stijn Goedertier, Jochen De Weerdt, David Martens, Jan Vanthienen, and Bart Baesens. Process discovery in event logs: An application in the telecom industry. *Applied Soft Computing*, 11(2):1697–1710, 2011. ISSN: 1568-4946. The Impact of Soft Computing for the Progress of Artificial Intelligence. doi:https://doi.org/10.1016/j.asoc.2010.04.025.

[26] Zhaoyang He, Yuyue Du, Lu Wang, Liang Qi, and Haichun Sun. An alpha-fl algorithm for discovering free loop structures from incomplete event logs. *IEEE Access*, 6:27885–27901, 2018. doi:10.1109/ACCESS.2018.2840818.

[27] IPBrick. Ipbrick, 2021. Accessed: 2022-03-12.

[28] IPBrick. Document management - iportaldoc, 2022. Accessed: 2022-03-12.

[29] Sander Leemans, Dirk Fahland, and Wil Aalst. Discovering block-structured process models from event logs - a constructive approach. pages 311–329, 01 2013. doi:10.1007/978-3-642-38697-8_17.

[30] Sander J.J. Leemans, Erik Poppe, and Moe T. Wynn. Directly follows-based process mining: Exploration a case study. pages 25–32, 2019. doi:10.1109/ICPM.2019.00015.

[31] Crosstab LLC. Streamlit review and demo: Best of the python data app tools. Accessed: 2022-09-01.

[32] A. Medeiros, A. Weijters, and Wil Aalst. Genetic process mining: An experimental evaluation. *Data Mining and Knowledge Discovery*, 14:245–304, 04 2007. doi:10.1007/s10618-006-0061-7.

[33] Wenyu Peng, Zhenyu Zhang, Ryan Hildebrant, and Shangping Ren. Empirical studies of three commonly used process mining algorithms. pages 2492–2499, 2021. doi:10.1109/SMC52423.2021.9658861.

[34] Portainer, 2022. Accessed: 2022-09-10. [link].

[35] Amazon Web Services. What is docker?, 2022. Accessed: 2022-09-10.

[36] A. Weijters, Wil Aalst, and Alves Medeiros. *Process Mining with the Heuristics Miner-algorithm*, volume 166. 01 2006.

[37] Fitri Yasmin, Faiza Bukhsh, and Patrício Silva. Process enhancement in process mining: A literature review. 12 2018.

MSc
2
CIC
FC
AN
U.PORTO
Online Process Extractor and Updator
Sónia Rafaela Costa da Rocha
FC

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

MSc