Real-Time Business Process Recommendations

João Miguel Valentim Rodrigues

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor(s): Pedro Manuel Moreira Vaz Antunes de Sousa
                José Carlos Paiva Rodrigues

Examination Committee
Chairperson: José Carlos Alves Pereira Monteiro
Supervisor: Pedro Manuel Moreira Vaz Antunes de Sousa
           António Manuel Ferreira Rito da Silva

October 2018
Acknowledgments

I’d like to dedicate a few words to the people that accompanied me throughout this journey. To my coordinators, Professor Pedro Sousa and Engineer José Rodrigues, I thank them, for giving me the opportunity to work with them, guiding me through the process of this dissertation, sharing their knowledge and expertise.

I would like to thank my family, for always supporting, caring and providing for me throughout all my academic journey, they were always there for me and without them, none of this was possible.

I would also like to thank my girlfriend Mariana, who always supported me and cared for me through thick and thin, she was a cornerstone and someone I could always count on for everything.

A thank you to everyone in Link Consulting Sa. for giving me the opportunity to work with them. A special thanks to Etelvina Cunha, who was not my coordinator but I saw as one.

A special thanks to all my friends, especially to the ones that I’ve known since my high school years and still accompany me today and to the ones that I’ve met through my academic journey, thanks to them, these rough years turned out to be the best years.
Resumo

Os processos de negócio possuem um grande número de decisões operacionais durante a sua execução. A lógica de decisão nestes pontos de decisão muitas vezes não é explícita ou optimizada e os intervenientes que executam estas atividades, podem encontrar dificuldades, que podem originar erros ou problemas de eficiência. Soluções nas áreas de Mineração de Processos e de Decisões abordam este problema, focando-se na descoberta de pontos de decisão e na representação explícita da lógica de decisão, num ambiente offline, majoritariamente para fins de gestão e análise de processos. No entanto, a Mineração de Processos e de Decisões pode também ser aplicada num ambiente online, de modo a oferecer suporte operacional (por exemplo, suporte à decisão), com o objectivo de apoiar e gerir instâncias de processos que estão a decorrer. Esta dissertação recorre aos logs de eventos criados por um processo de negócio real com o objectivo de descobrir os seus pontos de decisão e fornecer recomendações de "Melhor Acção Seguinte" aos intervenientes no processo que se encontrem em situações de indecisão nos pontos de decisão. Para tal, a solução transforma os pontos de decisão em problemas de classificação e aplica modelos de aprendizagem automática que aprendem e prevêm quais as melhores decisões a tomar em cada situação. A solução desenvolvida é avaliada e comparada com os dados do caso de estudo, mostrando resultados promissores.

Abstract

Business processes entail a large number of decisions during its execution. The decision logic in these decision points is often not explicit or optimized and might leave the process actors in an indecision situation, potentially leading to errors and inefficiencies. Solutions in Process Mining and Decision Mining have tackled this issue, focusing on discovering and explicitly representing the decision logic, in an offline setting, for management and analysis purposes. However, Process Mining and Decision Mining can also be used in an online setting, offering Operational Support (e.g. decision support), in order to support and manage ongoing process executions. This dissertation presents a semi-automatic solution aimed at providing a real-time recommendation system. This solution uses the event logs created by a deployed business process to discover its decision points and provide real-time "Next Best Action" recommendations to the business process actors who find themselves in an indecision situation in the discovered decision points. To do so, it turns the decision points into classification tasks and applies machine learning algorithms to learn and predict the best actions in each situation. The developed approach is evaluated with the event logs and compared with other solutions in this area, showing promising results.

Keywords: Process Mining, Decision Mining, Event Logs, Machine Learning, Recommendation System, Operational Support, Decision Points.
Contents

Acknowledgments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . iii
Resumo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . v
Abstract . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . vii
List of Tables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xi
List of Figures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiii
Nomenclature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
Glossary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1 Introduction 1
1.1 Problem Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
1.2 Objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
1.3 Case Study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.4 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
1.5 Thesis Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.6 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2 Related Work 9
2.1 Process Mining . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
2.2 Decision Mining . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.3 Recommendation Systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

3 Solution 13
3.1 Decision Point Identification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
3.2 Turning Decision Points into Classification Tasks . . . . . . . . . . . . . . . . . . . . . . . . . 14
3.3 Classification Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
  3.3.1 Hybrid Naïve Bayes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
3.4 Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
  3.4.1 Event Logs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
  3.4.2 Event Logs Filters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
  3.4.3 Decision Point Miner . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  3.4.4 Decision Point Filter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  3.4.5 Footprint Miner . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.6</td>
<td>Footprint Filter</td>
<td>20</td>
</tr>
<tr>
<td>3.4.7</td>
<td>Train and Evaluate Classifiers</td>
<td>21</td>
</tr>
<tr>
<td>3.4.8</td>
<td>Classifiers</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>Decision Point Results</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>Footprints Results</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>5.1 Notify Repair State</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>5.1.1 Feature Analysis</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>5.1.2 Class Analysis</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>5.2 Debit Notification Validation</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>5.2.1 Feature Analysis</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>5.2.2 Class Analysis</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>5.3 Responsibility Validation</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>5.3.1 Feature Analysis</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>5.3.2 Class Analysis</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>5.4 Communicate DoA Advise</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>5.4.1 Feature Analysis</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>5.4.2 Class Analysis</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>Classifiers Results</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>6.1 Evaluation Metrics and Methods</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>6.2 Evaluation Results</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>6.2.1 Notify Repair State</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>6.2.2 Debit Notification Validation</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>6.2.3 Responsibility Validation</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>6.2.4 Communicate DoA Advise</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>6.2.5 Overall Comparison</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>7.1 Achievements</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>7.2 Future Work</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>7.3 Communication</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td><strong>Bibliography</strong></td>
<td>49</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Control Perspective ................................................. 4
1.2 Data Perspective .................................................. 4

3.1 Example of an activity with two different outcomes ...................... 13
3.2 Footprint Table Excerpt for decision point Debit Notification Validation ........ 15
3.3 Recommendation Example .......................................... 18

6.1 Binary Confusion Matrix .......................................... 37
6.2 Multi-class Confusion Matrix ..................................... 38
List of Figures

3.1 Recommendation Request ................................................. 17
3.2 System Response ............................................................. 17
3.3 Solution Architecture ....................................................... 18
3.4 UML Class Diagram ......................................................... 19

4.1 Number of each type of Decision Points ................................. 24
4.2 BPMN model regarding the activity Responsibility Validation .......... 25

5.1 Warranty Budget Feature Values Distribution .......................... 27
5.2 Own Brand Feature Values Distribution .................................. 27
5.3 Repair Name Feature Values Distribution ............................... 28
5.4 Notify Repair State Outcome distribution ............................... 28
5.5 Contestation Motif Values Distribution .................................... 29
5.6 roleENT Values Distribution ............................................... 30
5.7 Amount Values Distribution ............................................... 30
5.8 Cause Values Distribution .................................................. 31
5.9 Debit Notification Validation Class Distribution ....................... 31
5.10 Entity Type Distribution .................................................... 32
5.11 Entity ID Distribution ....................................................... 33
5.12 Amount Distribution ......................................................... 33
5.13 Responsibility Validation Class Distribution ........................... 34
5.14 Communicate DoA Advise Class Distribution .......................... 35

6.1 Classifier Comparison for Decision Point Notify Repair State ........ 40
6.2 Hybrid Naïve Bayes Confusion Matrix for Decision Point Notify Repair State 40
6.3 Classifier Comparison for Decision Debit Notification Validation .......... 41
6.4 Hybrid Naïve Bayes Confusion Matrix for Decision Debit Notification Validation 41
6.5 Classifier Comparison for Decision Responsibility Validation ............ 42
6.6 Hybrid Naïve Bayes Confusion Matrix for Decision Responsibility Validation 42
6.7 Classifier Comparison for decision point Communicate DoA Advise .......... 43
6.8 Hybrid Naïve Bayes Confusion Matrix for Decision point Communicate DoA Advise 43
6.9 Classifier Comparison between all the classifiers ........................ 44
Chapter 1

Introduction

Business Processes are at the core of any organization, and organizations often deploy some sort of Business Process Management (BPM) engine in order to model, control and manage its processes. The execution of deployed processes leaves a trail of execution created and saved by said BPM engines. The trails are commonly called event logs and enabled a whole new discipline and area of research called Process Mining.

Process Mining is the discipline that takes advantage of the process’ event logs in order to extract useful information capable of providing a deep insight into the process’ behavior. Process Mining solutions can be categorized in four main categories: Process Discovery, Process Conformance, Process Enhancement and Operational Support. Various approaches and solutions have been proposed and developed, which have been implemented in real-world scenarios, where it was proven how useful the event logs are, when taken into consideration in the process engineering procedure. In this dissertation we tackle an issue in the field of Process Mining, concerned with the decisions encompassed during the execution of business processes.

Process Mining has a lot of applications, useful to business process managers and to the companies that apply it to their processes. It is common to segment process mining approaches in different perspectives. The main perspectives, are:

- Control-Flow perspective, concerned with the discovery of the order of the activities in a process;
- Organizational perspective, concerned with the discovery of the organization and sometimes the hierarchy of people that intervene in the processes;
- Data-flow perspective, a perspective mainly focused on the analysis and use of the data of the process;
- Time perspective, which uses the time information of the process, its activities and transitions to extract knowledge;
- Conformance perspective, concerned with verifying if the process is running as it was modelled.

These perspectives are enabled by the event logs, different event logs with different information allow different solutions to focus on different perspectives. Solutions in Process Mining, therefore, are focused on exploring one or multiple perspectives.
Business processes are not linear and more often than not, business actors will have to face decisions where many courses of action are available. The decisions might be influenced by a variety of factors, some are dependent on the process instance data. However, it can also rely on the business actor’s experience and knowledge about the process at hand. In a management point of view, one would prefer that the decisions taken during the process would only rely on the process instance and not on the actor since two different actors might take two different courses of actions under the same conditions and inevitably lead to different process outcomes and performance measures.

**Decision Mining** can be applied to **Process Mining**, which from the event logs aims at deriving decision points and decision logic explaining under which circumstances one course of action is preferable to another.

To find the decision rules, **Decision Mining** techniques must identify decision points in the process, find their features (expressed by the process data) and apply some machine learning algorithm (e.g. Decision Trees, Naive Bayes, Clustering, etc.). Although there is already a wide range of research conducted in the field of Decision Mining applied to business processes, this was mainly focused on discovering and representing decision logic, in order to provide process insight and to annotate the decision point with the tacit decision logic. However, these concepts can also be applied in an online recommendation setting, aimed at providing a "Next Best Action" prediction to a business actor that is unsure as to which action to perform. This can help organizations turn their processes into more agile procedures, that learn the most fitting procedures from historical executions.

In more recent studies, the concepts of recommendation systems and real-time operational support has been gaining relevance with authors recognizing that Process Mining and Decision Mining applied to an "online scenario" is possible and may bring relevant benefits to organizations, e.g. in the form of recommendation systems.

Given this context, we will now propose an approach that provides real-time recommendations, ranks the recommendations in terms of its probability and learns continuously from new observations. To do so, we will use novel decision mining techniques and a probabilistic classifier (Naïve Bayes). What drives this idea is the will to provide to the business actor a real-time recommendation that is based on historical event logs, taking into account the process instance data, to achieve a cooperation between the BPM system upon which the process is deployed and its actors.

The solution proposed was implemented using a set of event logs from an established process from one of the biggest retailing companies in Portugal. However, the concepts, approaches and algorithms can be applied in other domains.

### 1.1 Problem Description

Process mining covers a wide range of analysis tools and methods that can provide valuable insight into a deployed process. Decision mining applied to business processes in particular, has been mainly focused on discovering and representing decision logic. However, as pointed out in [3], "process mining should not be restricted to off-line analysis and can also be used for on-line operational support. Three operational support activities can be identified: detect, predict, and recommend". This thesis project lies on this field of operational support and will predict and recommend actions.

Business processes carry a variety of decisions that influence its execution and outcome. If no formal decision
criteria is available, these decisions are influenced by the personal knowledge of the person who decides and, therefore, are subject to errors that can lead to inefficiencies.

As such, this project aims at providing real-time recommendations to help decision makers avoid wrong decisions, which can result in better process performance. We will also be able to provide case performance predictions, that will be useful to determine when a certain case will be finished. These capabilities will be enabled by the analysis of historical and real-time event logs, bridging the gap between off-line and on-line analysis and support.

Example: Following the scope of the process that was worked on, consider a device repair process at a retailer. When a client provides its equipment to the clerk, an evaluation and budget are made and a decision has to be taken. The device can be repaired in the retailer’s repairing facilities, it can be sent to the supplier or it can be substituted. When taking this decision, a lot of factors need to be taken into account. For example, the age of the device, the supplier, the budget for the repair, the price of a new equal device, among others. As such, it requires a thorough evaluation by the clerk.

The evaluation and subsequent decision might be influenced, not only by the data of the process instance but also by the clerk’s experience. An optimal decision might not be taken or might take some time to make, resulting in possible delays, higher costs and unnecessary activities being executed. For example, if a substitution is the best action given the data available to the clerk, but the equipment is sent to the supplier, the process execution will be delayed and the cost, to the retailer, of repairing the device might be higher than the price of substitution. If the clerk was able to ask the system for a recommendation, the system would have access to the process instance data and use it to infer what the best action is. To do so, an a priori evaluation (learning phase) of historical data was already carried out and a prediction (prediction phase) with the ongoing process data is made. The possible actions and its probability would be promptly provided, helping the clerk make a more informed decision and avoiding inefficiencies.

1.2 Objectives

The main objective of the solution proposed in this document is to provide a Probabilistic ”Next Best Action” recommendation to the business users in order to help them make decisions during the execution of the business processes. These recommendations must be ranked in terms of their likelihood of being the ”Next Best Actions”.

One added objective of this project is that the algorithms must continuously learn from new observed executions.

To do so, the solution must introduce Machine Learning capabilities, that learn the past from the historical event logs and predict the most fitting decisions in order to provide the user a recommendation based on the control-flow and data-flow of historic and real-time event logs.

Therefore, the solution must be composed of two main components the Machine Learning component and the Recommendation System component, each with its associated objectives. The Machine Learning component is responsible for the learning from the historic event logs and continuously learning from the real-time logs, and the Recommendation System component is responsible for predicting the recommendations, ranking them and providing them to the business process actors.
1.3 Case Study

The solution follows a methodology that can be applied in multiple environments. However, in the scope of this dissertation, we are focusing on a specific case. We are working with the event logs of one of the major retail companies in Portugal, and these event logs are relative to its device repairing process. This process starts when a client delivers a device for repair and ends when said device is delivered back to the client, repaired or substituted. The device might be repaired in the retailer’s store or factory, might be sent to the supplier, might be promptly substituted or might be repaired at the clients residence. The devices span across multiple types (e.g. home appliances, cellphones, laptops, etc.), might belong to different suppliers, and many other nuances that influence the process.

This process is deployed since 2003, in multiple locations across the country and in Spain and has experienced some changes since then. Therefore, we are working with the data of finished process instances from 2015, so the data is similar to the present state of the process. The execution and management of process instances is carried out by multiple actors, with the help of a BPM Engine. The historical event logs were provided by a Oracle™ database that stores finished process instances.

These event logs have two perspectives. The Control Perspective (with the information about activity names and execution times) and the Data Perspective (which contains the data-objects of the process, also known as payload). However, the solution can be configured in other environments, if the data has the canonical form defined in Tables 1.1 and 1.2.

Table 1.1: Control Perspective

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Activity ID</th>
<th>Task Number</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>213111</td>
<td>Analyze Budget</td>
<td>213</td>
<td>Approve</td>
</tr>
<tr>
<td>213111</td>
<td>Deliver Device</td>
<td>214</td>
<td>Delivered</td>
</tr>
<tr>
<td>123400</td>
<td>Analyze Budget</td>
<td>401</td>
<td>Reject</td>
</tr>
</tbody>
</table>

Table 1.2: Data Perspective

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Data Object ID</th>
<th>Data Object Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>213</td>
<td>Amount</td>
<td>120.00€</td>
</tr>
<tr>
<td>213</td>
<td>Own Brand</td>
<td>Yes</td>
</tr>
<tr>
<td>401</td>
<td>Amount</td>
<td>80€</td>
</tr>
<tr>
<td>401</td>
<td>Own Brand</td>
<td>No</td>
</tr>
</tbody>
</table>

The control perspective table is the one needed to mine the process’ control-flow perspective and to discover the decision points. The data perspective is used to discover the process’ data-flow and the activities’ features. One important feature of these event logs is the Outcome column in the control perspective, since this gives us the information about the decision of the process actor at the time he/she executed that activity.

One important thing to note is that the event logs were not validated by anyone in the process. Meaning that, the event logs were created by the execution of the process and that is the state in which we received them. Therefore, it might contain errors performed by the actors, that are impossible for us to detect or errors produced by the logging system, which might be easier to spot (e.g. missing data). The algorithms that we developed worked with this data and the machine learning classifiers learned from this data as well, which in turn might influence the results.

Usually, in machine learning projects and solutions, the data is produced and verified by someone with deep
knowledge on the subject, in order to guarantee its veracity. For example, the well-known Iris Flower Dataset\(^1\), which is widely used in machine learning demonstrations and applications, was produced by a statistician and biologist and therefore it can be said to be trustworthy.

However, in order for this solution to be developed, we have to assume that, in the majority of the times, the business process actors made the right decisions and that the data was well logged by the system. The cases where the errors can be spotted were fixed accordingly.

### 1.4 Background

In order to better understand the topics in this dissertation, relevant concepts must be introduced. This section discusses some of the terms that we will be used throughout the document and are widely used in the Process Mining field of study.

- **Event Logs**: Event Logs in the BPM domain refer to the data recorded (logged) during the execution of the business processes by some Process Aware Information System (PAIS). This data provides valuable insight into each process instance and how it was actually carried out, which can deviate from the hand modeled process since it is not uncommon to encounter a number of cases that cannot be fully replayed by the model from beginning to end. Event logs may contain more or less features. However, there must be a sub-set of available features that are crucial to perform Process Mining techniques, in particular: the Case ID of the process instance, the executed Activity and some form of timestamp (e.g.: duration, start date, end date). Whenever possible, process mining techniques should use extra information. Each entry of the event logs represents an execution of an activity, often called task. Each task has an unique Task Number.

  Given an Event Log \( E \) and two tasks \( x \) and \( y \) then:

  \[
  \forall x, y : x \in E \land y \in E \land x \neq y \Rightarrow x.TaskNumber \neq y.TaskNumber
  \]

- **Traces**: Each trace describes the life-cycle of a particular process instance, also called case, hence the Case ID notion. Each trace represents 1 or more entries on the event logs, where all the entries corresponding to a trace have the same Case ID and for any 2 distinct given traces their Case ID is also distinct.

  Given a set of traces \( T \), then:

  \[
  \forall x, y : x \in T \land y \in T \land x \neq y \Rightarrow x.CaseID \neq y.CaseID
  \]

  On the other hand, all the tasks (activity executions) inside a Trace have the same Case ID.

- **Partial Traces**: Like the traces, however these describe the life-cycle of a process instance up to this time, which means that they represent on-going cases.

- **Process Data Objects (PDOs)**: Data objects specific to the process. This is the extra data that might be available in the event logs or in some other data source. PDOs are changed during the execution of the process.

\(^1\)https://archive.ics.uci.edu/ml/datasets/iris
• **Activity Footprint**: An Activity’s Footprint is the state of the activity at a given time. This state is composed of the data logged and the values of the PDOs. Each activity might change or use the value of 0 or more PDOs. The values of the PDOs at the time that an activity was performed represents this activity’s footprint. This concept is very important, since the activity footprint in a given process instance can give valuable insight as to why and how the activity was executed and why the succeeding activities were performed.

1.5 **Thesis Outline**

In this document we will start by presenting the Related Work in Chapter 2 and how it contrasts with our solution in Chapter 3. In Chapters 4, 5 and 6 we present the results and its evaluation, focusing on the analysis and comparison with other solutions, where a comparison is possible. In Chapter 7 we end this document with a conclusion, reflecting on our solution, achievements and Future Work.
1.6 Background

In order to better understand the topics in this dissertation, relevant concepts must be introduced. This section discusses some of the terms that will be used throughout the document and are widely used in the Process Mining field of study.

- **Event Logs**: Event Logs in the BPM domain refer to the data recorded (logged) during the execution of the business processes by some Process Aware Information System (PAIS). This data provides valuable insight into each process instance and how it was actually carried out, which can deviate from the hand modeled process since it is not uncommon to encounter a number of cases that cannot be fully replayed by the model from beginning to end. Event logs may contain more or less features. However, there must be a sub-set of available features that are crucial to perform Process Mining techniques, in particular: the Case ID of the process instance, the executed Activity and some form of timestamp (e.g.: duration, start date, end date). Whenever possible, process mining techniques should use extra information. Each entry of the event logs represents an execution of an activity, often called task. Each task has an unique Task Number.

  Given an Event Log $E$ and two tasks $x$ and $y$ then:

  \[ \forall x, y : x \in E \land y \in E \land x \neq y \Rightarrow x.TaskNumber \neq y.TaskNumber \]

- **Traces**: Each trace describes the life-cycle of a particular process instance, also called case, hence the Case ID notion. Each trace represents 1 or more entries on the event logs, where all the entries corresponding to a trace have the same Case ID and for any 2 distinct given traces their Case ID is also distinct.

  Given a set of traces $T$, then:

  \[ \forall x, y : x \in T \land y \in T \land x \neq y \Rightarrow x.CaseID \neq y.CaseID \]

  On the other hand, all the tasks (activity executions) inside a Trace have the same Case ID.

- **Partial Traces**: Like the traces, however these describe the life-cycle of a process instance up to this time, which means that they represent on-going cases.

- **Process Data Objects (PDOs)**: Data objects specific to the process. This is the extra data that might be available in the event logs or in some other data source. PDOs are changed during the execution of the process.

- **Activity Footprint**: An Activity’s Footprint is the state of the activity at a given time. This state is composed of the data logged and the values of the PDOs. Each activity might change or use the value of 0 or more PDOs. The values of the PDOs at the time that an activity was performed represents this activity’s footprint. This concept is very important, since the activity footprint in a given process instance can give valuable insight as to why and how the activity was executed and why the succeeding activities were performed.
Chapter 2

Related Work

Over the last years, the topics of Process Mining and Decision Mining have been gaining relevance and valuable research has been conducted in the area. This was mainly fueled by the growing popularity of Enterprise Resource Planning systems (ERP-systems) and other PAIS. In this section, a literature review on the subject is introduced, mainly focusing on the most relevant topics: Process Mining, Decision Mining and Recommendation and Prediction Systems.

2.1 Process Mining

In [2] the author identifies the main capabilities of the process mining "toolbox":

- Automatically discovering processes without any modeling,
- Finding bottlenecks and their causes,
- Detecting and understanding deviations, to measure their severity and to assess the overall level of compliance,
- Predicting costs, risks and delays,
- Supporting redesign,
- Recommending actions to avoid inefficiencies.

The main fields of process mining were identified by A. Rozinat cf. [1]:

- **Process Discovery**: The discovery of business process models based on the logs created by the deployed processes to be discovered;
- **Process Conformance**: Check if the deployed processes are in conformance with the hand modeled processes;
- **Process Enhancement**: Enhancing the business process model specification with key indices analyzed from the logs. For example, Time measures, bottlenecks, etc…
These three fields were widely accepted and researched by the community, c.f [1, 2, 8, 15–19, 21]. However, in more recent articles, in particular in [2], the author identifies a fourth field, the **Operational Support** field. In contrast to the previous fields, this encompasses tools that provide real-time support to the process execution. The tools that constitute the previous fields are focused on providing insight on the process in an offline setting.

In [15] the authors go over the subject of Process mining defining it as “the method of distilling a structured process description from a set of real executions”, illustrate its potential but also its scientific challenges that need to be addressed. The challenges identified in this article, are also acknowledged by most of the literature on this topic. The most relevant are:

- **Mining Invisible Tasks:** When certain activities of a process are not recorded in the event logs.

- **Mining duplicate tasks:** Challenge referring to the situation where one can have a process model where two distinct nodes are referring to the same task.

- **Mining Loops:** When in a process execution flow it is possible to execute the same task multiple times.

- **Dealing With Noise:** The logs may contain incorrectly logged data, commonly called “noise”. Most algorithms assume the data is correct, and therefore the unexpected inconsistencies can cause errors in the process mining procedure. However, probabilistic and stochastic approaches, c.f. those presented in [17] with Markovian models, can deal with noise.

- **Dealing with incompleteness:** When a log doesn’t contain sufficient information to derive the process.

- **Gathering data from heterogeneous sources:** Most PAIS are capable of logging event data, however, the format of this data is specific to the system in use. Attempts have been made to create a standard representation of logs for process mining, c.f. MXML\(^1\) (Mining XML) and XES\(^2\) (eXtensible Event Stream).

In [20], the authors define several *process perspectives* that need to be correlated in the process mining implementation. These perspectives are:

- The **control-flow** perspective (e.g., the next activity to be performed)

- The **data-flow** perspective (e.g., type of equipment, equipment supplier)

- The **time** perspective (e.g., the activity duration)

- The **resource/organization** perspective (e.g., the resource that performed the activity)

- The **conformance** perspective, if a normative process model exists (e.g., the execution of activities in the wrong order)

When performing process mining techniques, one can take into account one or more of these perspectives. Also in [20], the authors apply methods of *log manipulation* to enrich the event logs before applying the machine learning algorithms. After this, *decision tree regression* algorithms are applied to classify and cluster the event logs into chunks of similar event logs. An important concept used is the idea of splitting the traces, creating *sub-logs*, which create partial process models that are easier to interpret.

---

1 http://www.processmining.org/logs/mxml
2 http://www.xes-standard.org/
2.2 Decision Mining

In [1] the author defines the concept of Decision Mining in the context of Process Mining as the application of data mining techniques for the detection of frequent patterns in business processes, providing valuable insight into the process and making tacit knowledge explicit. In this book, Anne Rozinat identifies the two major steps for deriving the decision logic of a process from its event logs.

1. First, the decision point must be identified. The author identifies a decision point as a place with multiple outgoing arcs. That is, an activity that, in different traces (process instances), has two or more distinct successors.

2. Second, the decision point needs to be turned into a classification task. In this classification task, the classes are the different decisions that can be made, and the attributes used for classification are the data values.

The classification algorithm used is the Decision Trees Classification, where for each decision point there is an associated decision tree. This classification method is also the one chosen in most of the implementations on this topic, c.f. [2, 7–11]. In this book, a plug-in for the tool ProM is presented, the Decision Miner plug-in. Given the event logs in a canonical form, the tool discovers the underlying process model and the decision trees for each of the decision points identified. The author also identifies the main challenges inherent to the decision mining process. First, the usual challenges related to supervised learning, such as noise in the data, incomplete training sets and over-fitting. Second, the challenges related to Process Mining, such as invisible tasks, duplicate tasks and loops.

In [9] the authors propose a novel approach for decision mining based on alignments. The authors present a way of aligning an event log with data and a process model with decision points. These alignments can be used to generate a well-defined classification problem per decision point. The authors use Petri Nets discovered with the control-flow information and enrich it with the data-flow information, creating a Petri Net with Data.

2.3 Recommendation Systems

The solution proposed in [12] provides the users a recommendation service based on the control-flow perspective. It uses the historical traces provided in the event-logs to predict what the next action should be. To do this, the system matches the user’s partial trace, i.e. that belongs to the running process, and matches it with the historical traces. Then, each trace ”votes” on what the next action should be. The more similar the trace is to the partial trace the more its ”vote” counts. So, this is a recommendation system based solely on the sequence of actions. The solution proposed on this document has a similar approach in terms of the final objective, which is providing real-time recommendations. However, our solution takes into account the process payload. Therefore, the recommendations are be better suited for each recommendation request, since they are based on the process data and not only on the sequence of actions.

In [7] a semi-automatic approach is proposed. This solution improves the business performance of processes by learning and deriving decision criteria formulated as decision rules from the experience gained through past

3http://www.processmining.org/prom/start
process executions. This recommendation system uses decision mining, decision tree learning and path finding on the decision trees to determine which paths lead to the best outcomes. To allow for this notion of outcome ranking, the system uses the notion of process hard goals and process soft goals, which are, respectively, the process termination states and the process performance indicators. To determine these measures, a significant amount of process specific knowledge is necessary. The learning procedure only considers the process instances that reach desirable end states, ignoring instances that, for example, reached exceptions, even if the deviant behavior has a significant amount of instances. This detail differs from our approach, since we take into account all the process cases that we have at our disposal, therefore the system learns from all kinds of process cases and not only those that ended in desireable states.

In [13] the authors address the problem of recommending activity steps in collaborative IT support Systems by automatically discovering and annotating process models and with the introduction of a recommender. To do so, the authors developed a solution that analyses past case executions, discovers the step flow model, annotates it with case metadata and uses the metadata in open cases to match it with the annotated model and recommend the best next actions. In this solution, the authors opted for a more model-centered solution, whereas in our solution we are only focusing on the activities that were identified as decision points and the process payload in those activities, applying machine learning capabilities in said decision points.

The solution presented in [14] is also focused on the application of process mining to operational decision making, presenting a generic framework and a ProM plug-in for operational support. The solution uses the time information to check the execution times of running cases and recommends actions that achieve better times. In our solution, as explained before, there isn’t such a great focus on the process itself as a whole, but mainly on the activities that are decision points. Also, our solution, takes into consideration more information about the activities behavior, by using the data objects to apply machine learning techniques and recommend the best suited actions.
Chapter 3

Solution

In this section, we present the solution formulated through the analysis of the use case and of other solutions in the area. There are 3 main solution modules: the identification of decision points, how to turn those decision points into classification tasks and the classifiers chosen and implemented.

3.1 Decision Point Identification

As pointed out in [1], the first step in decision mining is to identify the decision points in the process. To do so, one must analyze the event logs and identify possible places where more than one action is possible. Depending on the event logs, these possible actions may differ. For example, in our case study, the actions performed by the business actor are described in the "Outcome" data object of the activities. However, in other cases, the possible actions in an activity might be the succeeding activities that are executed, as is presented in [1]. In this book, the author defines a succeeding activity as an outgoing arch.

To identify the decision points, we look at every instance of every activity in the event logs, capture the set of its outcomes and if the set has more than one outcome for an activity, that activity is flagged as a decision point. Consider Table 3.1 as an example. In this event log excerpt, there are two CaseID’s, meaning that the two rows belong to different process instances. In these two different process executions, the activity Analyze Budget, has two different outcomes, therefore, its outcome set would look like so: outcomeSet = \{Accept, Reject\}. It has two different outcomes, and therefore, the activity Analyze Budget is flagged as a decision point.

<table>
<thead>
<tr>
<th>CaseID</th>
<th>Activity Name</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>537132415</td>
<td>Analyze Budget</td>
<td>Accept</td>
</tr>
<tr>
<td>568075942</td>
<td>Analyze Budget</td>
<td>Reject</td>
</tr>
</tbody>
</table>

In our approach, by taking into consideration only the activity in itself to identify it as a decision point or not, it is not necessary to "look ahead" and see which activities come next, almost like a Markovian approach to the process.

Therefore we mitigated the issues related to decision point identification, laid out in [1], in particular challenges related to invisible tasks, duplicate tasks and tasks in loops, since these are only challenging if the approach
presented by the author is followed.

3.2 Turning Decision Points into Classification Tasks

After identifying the activities which are decision points, they need to be turned into classification tasks. A classification task, in the Machine Learning field, is a supervised or unsupervised learning problem where the output of the classifier is a discrete set of classes. Simply put, a machine learning classifier is a function that maps input data into a class. The input is a set of features and the class is what is supposed to be predicted, given the features.

Each decision point is treated as a singular classification task. Therefore, to turn the decision points into classification tasks, we need two things: features and classes. The features are the decision point’s payload in each execution and the classes are the outcomes.

The features are the data objects (payload) that are recorded every time an activity is executed. The data objects have different types, they can be numeric or textual. Therefore, to train a classifier on a decision point we need to extract its payload at each execution. There are three main ways to extract the data objects. Different methods were needed because of constraints in the case study’s data-set. In different process executions, the same activities have different data objects, therefore, extracting the features in a uniform way proved to be a challenge. However, we identified three different ways to perform the extraction, which are all viable depending on the event logs. They are:

1. Gather all data objects belonging to the decision point, that are present in each execution and keep just the ones that are present in all executions, i.e. the intersection between the sets of observed data objects. Since we observed that there was a great variance in data objects in some decision points, this, at first, seemed to be the best approach, since to apply machine learning capabilities, there should be a fixed set of features (data objects) that are present in every, or most, executions. This is also the best approach if we consider it in a operational point of view. Since the selected features are the ones that are present in all executions. And therefore, when a real-time recommendation request arrives, there is a high chance that all the necessary features are available for the recommendation system to work with;

2. Gather all data objects belonging to the decision point, that are present in each execution and keep all of them, i.e. the union between the sets of observed data objects. This was the main method used, since with the previous method, most decision points had only a small number of data-objects (sometimes even none). This method has issues, since it can happen that the selected data objects are not observed in all executions. However, a selection criteria can be applied, e.g. keep only the data objects that are observed in a certain percentage of the decision point’s executions. In our case we chose the data objects that were present in at least 75% of the decision point’s cases;

3. For each execution of the decision point, gather all the data objects belonging to the activities that were executed up until (and including) the execution of the decision point. This method depends heavily on the process and should only be applied if the previous methods aren’t able to gather a significant number of
Table 3.2: Footprint Table Excerpt for decision point Debit Notification Validation

<table>
<thead>
<tr>
<th>Constestation Reason</th>
<th>roleENT</th>
<th>Amount</th>
<th>Observations</th>
<th>Cause</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Delay</td>
<td>SCR</td>
<td>41.56</td>
<td>Debit Authorized by Insurer</td>
<td>Exchange Authorized</td>
<td>Reject Debit</td>
</tr>
<tr>
<td>NA</td>
<td>FORN,6008</td>
<td>23.73</td>
<td>Client in Store</td>
<td>Exchange Authorized</td>
<td>Accept Debit</td>
</tr>
</tbody>
</table>

features, since the number of selected data objects might be too big. This method is better suited in event logs where most activities have the same or a similar set of data objects;

After discovering the features, the data needs to be organized in tables, footprint tables, ready to be analyzed by the classifier. There needs to be one classifier per decision point, therefore, one table per classifier, where each row is a decision point’s execution and each column is a feature, being that in the last column we have the Outcome which is the only column that is present in every footprint table. An example of one of these tables is presented in Table 3.2.

### 3.3 Classification Algorithms

To learn from the data and to predict new instances, we chose and developed three classifiers:

- **Baseline**: the simplest classifier and the one that was developed first in order to create a benchmark. This is a probabilistic classifier, that always chooses the class that has the highest rate of execution. For example, if in a decision point, an outcome is observed 90% of the times, the baseline classifier for that decision point will always recommend that outcome;

- **Decision Trees**: standard Decision Tree classifier, as the one used in most implementations, as presented in Chapter 2;

- **Hybrid Naïve Bayes**: Naïve Bayes classifier, capable of handling both numerical and textual features, applying a Gaussian distribution to the numerical features and standard Bayes probability theory to the textual data. This algorithm works better if the all the features are independent. This is a prerequisite that was observed in the decision point’s data-set, since every feature relates to a different process data object.

To choose the main algorithm, we had to consider the requirements. The classifier has to be able to provide probabilistic insight about the possible actions, ranking them. Therefore, decision trees was not the core algorithm used, unlike most solutions in this area. The classifier must be fast when predicting the recommendations, since in a real-time process the system has to output the decision in a useful time window. And it must learn continuously from new observations. A solution meeting these requirements, differentiates itself from other solutions in this area, like the ones presented in Chapter 2.

Considering these requirements, we developed a Hybrid Naïve Bayes classifier, capable of learning in bulk and with new singular instances (continuous learning). Since it is a probabilistic classifier it can rank the possible actions in terms of their probability.
3.3.1 Hybrid Naïve Bayes

In our solution we developed an Hybrid Naïve Bayes with some adaptations to allow for smoothing. To implement Naïve Bayes, we assume that the various features describing the decision point are independent given the outcome. With this algorithm, it becomes easy to calculate the probability of the different outcomes for a specific instance, given the vector of its features.

Therefore, to calculate the probability of an instance I, with two possible outcomes o1 and o2, and the vector of features v = [value1, value2], the probabilities are given in the following way:

- For outcome o1:
  \[ P(o1|value1, value2) = P(o1) \times P(value1|o1) \times P(value2|o1) \]

- For outcome o2:
  \[ P(o2|value1, value2) = P(o2) \times P(value1|o2) \times P(value2|o2) \]

Where the conditional probabilities are given by, for example:

\[ P(value1|o1) = \frac{c(value1 \cap o1)}{c(o1)} \]

Where c(argument) represents the counter of its argument in the data provided.

The procedure is the same if more outcomes are possible. After calculating all the probabilities for all the outcomes, one can normalize them, since multiplying probabilities can lead do very small values.

With this approach we also developed a continuous learning environment in a seamless way. With each recommendation, the user can also input what its decision actually was and that information will be added to the information already available in the system. This way, there is no need to periodically input a large sample of data each time we want the system to learn again, promoting continuous learning.

To successfully use Naïve Bayes, one should also care about not zeroing probabilities. This can happen if an instance has unseen feature values in the training data, for example, if \[ P(value3|o1) = 0 \], when we calculate the probability \[ P(o1|value1, value2, value3) \] it will be 0, since we’re multiplying probabilities. This can lead to unwanted results, e.g. if value3 was not seen for any outcomes, all probabilities will be equal to 0. To solve this problem, one should smooth the probabilities to account for unseen features.

**Smoothing**

There are two main types of observed features, discrete and continuous. Both need to be smoothed, however different measures need to be taken to deal with both of these types.

For discrete features we performed a Laplace Smoothing. To do so we added 1 to every feature counter when calculating the probability. This way no counter will be zero, since if the value of a feature (data object) was never observed, its counter will be equal to 1, instead of 0 and therefore, the associated probability will have a small residual value. To prevent the probabilities summing to more than 1, we have to add the Volume (V) to the divisor.
The Volume is simply the number of different discrete values observed in the data for the feature being analyzed. Therefore, the probabilities become:

\[ P(value_1|o_1) = \frac{c(value_1) + 1}{c(o_1) + V} \]

As for continuous features, we approximated the possible values to a Gaussian distribution, calculating the \( \mu \) (mean) and \( \sigma \) (standard deviation).

**Behavior Example**

To turn this solution into a real-time system it can be provided as a web service integrated with the BPM Engine that runs the process. The behaviour of such system is exemplified in Figures 3.1 and 3.2. In this example, the system’s classifiers are already trained (Hybrid Naïve Bayes Classifiers). The training flow is not exemplified but it is simply a pipes and filters architecture with the steps explained in this section (decision point identification, turning decisions into classification tasks and training the chosen classifier).

![Figure 3.1: Recommendation Request](image)

![Figure 3.2: System Response](image)

To exemplify what the Classification Request and the Recommendation in Figures 3.1 and 3.2 look like, consider the classifier trained with the data in Table 3.2. A Classification Request looks like a row in Table 3.2 but with the **outcome** value missing, since that is what the process actor wants to know. Then, the classifier would calculate its predictions and recommend to the user a recommendation like the one in Table 3.3, where all the possible outcomes are listed with the associated probability. Then, the user inputs to the system what its decision actually was, and that information is fed back to the recommendation system, that will add that new decision to its observations and learn from it.
Table 3.3: Recommendation Example

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept Debit</td>
<td>23%</td>
</tr>
<tr>
<td>Reject Debit</td>
<td>77%</td>
</tr>
</tbody>
</table>

### 3.4 Architecture

In this section the project’s Architecture is presented in figure 3.3. In this architecture, presented in a "Pipes and Filters” UML Component Diagram 3.3, the training phase of the solution is specified and each component is described.

![Figure 3.3: Solution Architecture](image)

#### 3.4.1 Event Logs

The training phase starts with the raw historic Event Logs, provided by the BPM engine. The Event Logs come in csv form and their general structure is presented in Chapter 1.3.

#### 3.4.2 Event Logs Filters

First, the Event Logs need to be carefully analyzed and filtered. As explained in 1.3, the Event Logs have two different perspectives, the Control Perspective and the Data Perspective. We had access to one year worth of Control Perspective logs but only two months (January and February) worth of Data Perspective logs, due to dimensionality issues, since each entry in the Control Perspective may have up to one hundred entries in the Data Perspective. Our solution is based in these two perspectives, they need to be matched and therefore we had to mainly work with two months worth of data.

As the data comes in raw, it needs to be carefully analyzed and filtered to deal with missing or erroneous information. Therefore, first, the Event Logs pass through a variety of filters. The first filters focus on the Control Perspective logs. After loading the file, the rows Task Number, Case ID, Activity Name and Outcome are selected since they are the ones needed in this and the following steps. If an entry has missing values in these rows it is filtered out, since they can’t be inferred and are crucial.

---

After filtering, the event logs are transformed into an object representation to better be dealt with. The Traces are a sequence of Tasks, which are instances of Activities with extra meta-information. However, the tasks in the Control Perspective logs aren’t ordered, therefore, their order has to be inferred through the analysis of the Task’s Creation Date.

![UML Class Diagram](image)

Figure 3.4: UML Class Diagram

After creating this representation, we are ready to discover the decision points.

### 3.4.3 Decision Point Miner

The decision point identification is an important step and is thoroughly explained in Section 3.1. This component runs through the set of activities flagging as **decision points** the activities that have 2 or more **outcomes**.

### 3.4.4 Decision Point Filter

After the identification of the decision points, these need to be analyzed and filtered. This isn’t an automatic filtering, needs human input and is an iterative procedure. There were three main filtering criteria:

- **Decision made by client**: Because of the nature of the process, some activities that were flagged as decision points, actually weren’t so. In some cases the outcome of the activities might not be dependent on the process data itself but rather on the client. For example, the activity **"Wait for Client’s Decision"** was flagged as a decision point. However, the possible outcomes are **"Unreachable"**, **Budget Approved** and **Budget Denied**. One can conclude that the process actor is not in charge on this decision, leaving it to the client, therefore, decision points like this are not considered. This filtering was done with someone from Link Consulting Sa. that was involved in the implementation of the process who deeply knows the case study’s process and was able to give us feedback in the flagged decision points.

- **Not enough observations**: Some activities that were flagged as decision points didn’t have enough observations. With less observations there are less decision point instances for the algorithm to learn with, which can produce bad results due to a faulty learning procedure. We defined the threshold at 300 observations and any decision point with less than that was not considered. This can be solved with a higher volume of event logs, however that was not possible and was postponed for future work.

- **Not enough features**: After discovering and filtering the decision points, when creating and filtering the decision point’s footprints, we realized that some decision points had no features (or data objects), making their footprints incomplete and not good enough to work with. When that happens, the decision point needs to be ignored, since with no features, the classifier can’t learn properly. We set the minimum at least one
feature per decision point, which is not desirable, since one feature is usually not enough to work with, however, if we raised this minimum we would not have a substantial number of decision points to work with. This can also be solved with a higher volume of event logs, however that was not possible and was also postponed for future work.

This filter is necessary, not because of the approach, but because of the nature of the process itself. The approach finds all the decision points plus some more activities that are decisions but not for the process actor do take.

3.4.5 Footprint Miner

After discovering and filtering the decision points, we need to create and cure the datasets. This is an iterative process, the Footprint Miner, Footprint Filter and Train and Evaluate Classifiers modules are executed and adapted many times for each decision point to achieve the best results possible. The three different approaches presented in Section 3.2 were tested and the second approach was chosen. The first one showed poor results in finding good features for some decision points and the third approach took a lot of time and memory, making it impossible to use because of the amount of features was too big to work with. Therefore, with the second approach we reached a balance.

3.4.6 Footprint Filter

After finding the footprints (one for each decision point), they were carefully analyzed, performing an Exploratory Data Analysis (EDA), that will be detailed in Section 5, to determine its traits and flaws.

There are two filtering procedures that need to be carried: Feature Filtering and Observation Filtering. Feature filtering is a column-wise filtering while Observation Filtering is a row-wise filtering.

Feature Filtering

The features had to be filtered and selected in order to have a good subset of good features for each decision point, instead of having a big set of bad features.

These features are relative to the dataset we worked with, however, they cover the main types of features that can be encountered in process’ event logs and in other solutions. These types are:

- Continuous numerical features, e.g. budgets;
- Discrete numerical features, e.g. numerical ID’s, which are not good features for a classifier and should be filtered out;
- Discrete textual features, i.e. a set of possible textual values;
- Open text features, i.e. textual descriptions entered by the process actors. These features can be valuable if interesting information can be extracted. For example, if the feature is open but common traits can be observed.
In all decision points, some features were filtered out. The first filtering criteria was the one explained in method two. This method was applied to every footprint, however, the rest of the filtering was conducted case by case. Selecting the final features is an iterative procedure and there are many different techniques that can be used, cf. [4]. The main techniques applied were:

- Tried different combinations of features by forward and backward selection
- Applied dimensionality reduction techniques to open text features to extract patterns
- Dealt with Missing data. For numeric missing data, we filled in the missing values with the mean of all values for that feature. For textual data, in some cases we deleted the rows that had missing data in the feature being analyzed and in other cases we considered the missing values (see the "NA" value in feature "Contestation Reason" in Table 3.2) as valid values. This dual criteria was based on the behavior of the classifier, testing each criteria and choosing the one with the best results.

**Observation Filtering**

Even after selecting the preferable features, there were still some observations that didn’t comply with the norm or had to be deleted.

There were two filtering criteria when dealing with the rows:

- Rows with missing values in some columns (features) were excluded;
- In some cases, a high bias of classes was present. This means that a footprint had a much higher number of rows belonging to one outcome than to the others and a high bias in the dataset have been reported to hinder the performance of the classifiers learning from that dataset. In these cases, we applied a common method called **Undersampling** and is thoroughly explain in [23]. This method aims at balancing the number of rows of each class by deleting a portion of the rows belonging to the majority class.

### 3.4.7 Train and Evaluate Classifiers

This is the module where we use the footprints to train the classifiers and, subsequently, evaluate them. The evaluation method, metrics and results will be presented in Chapter 6. The evaluation results serve as input for the filtering criteria, and what happens next depends on the evaluation results.

If the results are good or if no other steps are possible, the learning procedure ends, and the classifiers are ready.

Otherwise, if the results are subpar, the process is repeated. We either filter the decision point or we work on the footprint itself (which was the most common next step).

### 3.4.8 Classifiers

When the classifiers reach a good evaluation or if further improvements are not possible, the classifiers can be exported and used to provide real-time recommendations to ongoing process instances, e.g. provided as a web-service.
When a recommendation request comes in, it can belong to any of the discovered decision points and only one classifier is responsible for predicting the most fitting recommendation. Therefore, there is a layer responsible for the routing of the recommendation request as well as mining the necessary features and its values for the classifier to work with, since the recommendation requests come in with all the process payload and the classifiers only work with the subset of features that were selected during the steps described above.
Chapter 4

Decision Point Results

We will now talk about the discovered decision points with the approach presented in Section 3.1. There are three phases in the decision point discovery procedure:

- Decision Points automatically discovered;
- Decision Points filtered due to the decision not actually being made by the business process actor;
- Decision Points filtered because of lack of observations or features

In total, there are eighty one different activities in the event logs. From these eighty one, our algorithm identified thirty six as decision points. These thirty six were carefully analyzed with a business expert and from this procedure, nineteen activities were chosen as truly being decisions in the hands of the business process actors.

After choosing this subset, we had to assert if the nineteen chosen activities had enough data to work with, both regarding the number of features and the number of observations, since these are two core requirements in order to apply machine learning capabilities.

In the end, after applying all these filtering criteria, we ended with eight decision points to work with. The decision points’ names and their outcomes are:

- **Notify Repair State**: Budget, Parts Request, Repaired, No Breakdown, Substitution, Exchange Article;
- **Debit Notification Validation**: Accept Debit, Reject Debit;
- **Responsibility Validation**: Debit Another, Impose Debt;
- **Situation Analysis**: Article in Destination, Physical Displacement
- **Passage to Budget Analysis**: Passage Approved Client, Passage Approved Others, Passage Denied;
- **Budget Analysis**: Budget Approved, Budget Denied;
- **Budget Error Checkup**: Not Repaired, Repaired, Fix at Brand, No Breakdown, Validate Budget;
- **Communicate DoA Advice**: Accept, Reject;
As we can see in figure 4.1, we have four binary decision points and four multi-class. Therefore we’ll have to develop four binary and four multi-class classifiers.

In order to evaluate if the discovered and chosen decision points were in fact decision points, we looked at the BPMN model of the process that created the event logs. One of the advantages of our solution is that it is independent from the BPMN model, we only look at the data itself and focus on the singular activities to determine if they are decision points or not, which is an interesting achievement and differentiates our solution from others in the area. However, in order to verify that our results are trustworthy, we decided to assert if the discovered and chosen decision points were correct. In order to understand how the validation was carried out, consider Figure 4.2, paying special attention to the activity Responsibility Validation. In this model, the activities performed by humans are the ones in green, while the ones in blue are performed by the system. Responsibility Validation was one of the activities that we identified as decision point. In this activity, the actor makes a decision, Debit Another or Impose Debit. Further ahead in the process, this decision is analyzed and results in a bifurcation of the process, where only one path is taken, and we know this because of the Response BPMN gateway which is a XOR gateway. XOR gateways only activate one path and the path that is activated depends on some variable in the process payload. In this case, we can see that the variable it depends on is the Outcome variable of the activity Responsibility Validation, by the text next to the outgoing arcs of the XOR gateway, where we can read Debit Another and Impose Debit, which are exactly the outcomes of activity Responsibility Validation. We can then confirm that this activity, which was considered a decision point by our method, is in fact so. In this section we only look at this activity in particular, but all the other discovered and chosen decision points were verified likewise.

Furthermore, we can argue that the methods used in other solutions, especially the ones that were presented in Chapter 2, would not consider this decision to be a decision point, since it only has one possible outgoing arc, after the activity itself. In fact, said methods, would only consider as decision point the activity Update Responsible Entity, which is a system activity and therefore does not need a real-time recommendation service.
Figure 4.2: BPMN model regarding the activity Responsibility Validation.
Chapter 5

Footprints Results

In this section we will go over four decision points and demonstrate its’ footprints, paying special attention to the features and outcomes, their distributions and how they were analyzed and in some cases, transformed. Dataset analysis and transformation is a very important step, especially in this case since we are creating the decision points’ datasets. The footprints presented here are the datasets’ final state and the results that are presented in Chapter 6 were achieved with these the datasets in the state that their are presented here. Before achieving this state, they all went through an iterative process where many a lot of features were filtered out. In some cases, like in the last footprint presented in this chapter, where the resulting dataset is very poor, both in observations and features.

5.1 Notify Repair State

This decision point’s footprint has 40705 entries, three features and six possible outcomes. Its features are Warranty Budget, Own Brand and Repair Name and its outcomes are Budget, Parts Request, Repaired, No Breakdown, Substitution, Exchange Article

5.1.1 Feature Analysis

The first feature in this dataset is Warranty Budget. It is a Discrete Textual Feature with two possible values, G and O. The distribution of values for this feature is highly skewed towards the G value. Its distribution can be observed in Figure 5.1, where a high imbalance can be observed, with the value G appearing in 98.8% of the observations.
The second feature is **Own Brand**. It is also a **Discrete Textual Feature** with two possible values, **YES** and **NO**. The distribution of values for this feature is well balanced, as can be observed in Figure 5.2.

![Figure 5.2: Own Brand Feature Values Distribution](image)

The third and last feature is **Repair Name**. As the previous two, it is a **Discrete Textual Feature**, however, this feature has many more possible values (192) with the values **"MICROMA SA"** being the most common with 7036 observations. This feature’s distribution is summarized in Figure 5.3, where we can observe the top four most common features and its frequency.

![Figure 5.3: Repair Name Feature Values Distribution](image)
In this footprint, we deleted all the rows that had missing values in any of these features. Before choosing these three features the footprint had five features, however, we excluded two features since they only had one possible value, and therefore don’t add nothing to the footprint.

5.1.2 Class Analysis

In this decision point there are six possible outcomes, Budget, Parts Request, Repaired, No Breakdown, Substitution and Exchange Article. The outcome frequency distribution is represented in figure 5.4. As is clear from the graph, the class distribution is highly skewed towards the class Repaired. Therefore, in order to balance this dataset, we performed an undersampling, by randomly deleting 50% of all rows which belonged to the class Repaired.
5.2 Debit Notification Validation

This decision point’s footprint has 11929 rows. Its features are Contestation Motif, roleENT, Amount, Observations and Cause. And it has two possible outcomes Accept Debit or Reject Debit.

5.2.1 Feature Analysis

The first feature in this decision point’s footprint is Contestation Motif, which is a Discrete Textual Feature with three possible values: No Delay, Article not Provided and NaN. This is one of those cases where identifying Nan as a feature value was actually beneficial for the classifier results, and therefore it was kept. This feature’s values distribution can be analyzed in Figure 5.5.

![Figure 5.5: Contestation Motif Values Distribution](image)

The second feature is roleENT, a Discrete Textual Feature with 146 unique values. This feature also had a lot of entries with the value Nan, however, these instances were deleted since the classifier showed better results when the value was not considered. The value distribution can be observed in Figure 5.6 with special emphasis on the most common 4 values.
The third feature is **Amount**, a **numerical** feature. The missing values in this feature were substituted with the mean of all the values and the distribution can be found in Figure 5.7, both in a box plot and in a distribution plot. The feature has a mean of 81.24, a 25 percentile of 23.71, a 50 percentile of 41.07 and a 75 percentile equal to the mean, 81.24. From the distribution we can also see that there are a lot of outliers, represented by the points past the upper extreme line, which shows a lot of variance and a not so well behaved distribution. In the distribution plot we can see that most values are concentrated between the values 0 and 200, but go up to 2000.

The forth feature is **Observations** and this is an **Open Text Feature**, and therefore we performed a dimensionality reduction. To do so, we observed the feature values in order to identify certain patterns. One of the patterns discovered was, for example, the values **Debit Authorized by Insurer** and **Client in Store**, which were often times between some other text information. Therefore, we extracted some patterns and represented the values with the patterns discovered. For example, consider the value "Debit Authorized by Insurer. **AJSilva". Clearly, the second part, containing the name of the clerk who entered the information, doesn’t add nothing to the value itself and is just noise, and so, "**AJSilva" can be removed, while keeping the "Debit Authorized by Insurer” pattern.

The fifth and final feature is **Cause** and is a **Discrete Textual Feature** with 69 possible values. For this feature, the rows with "Nan” values were kept, since the results were better while considering the "Nan” as a value. The distribution can be analyzed in Figure 5.8, where we can see the 4 mos common values with the ”Nan” value being
the forth. If we deleted the rows with the "Nan" value in this feature, we would loose a lot of information.

Figure 5.8: Cause Values Distribution

5.2.2 Class Analysis

The two possible outcomes, Reject Debit and Accept Debit, and its distribution is represented in Figure 5.9, where we can see a skewed distribution toward the class Reject Debit. However, as we’ll see in the Section 6, the classifier still showed good results and therefore we didn’t perform any under or over sampling technique.

Figure 5.9: Debit Notification Validation Class Distribution

5.3 Responsibility Validation

We’ll now focus on decision point Responsibility Validation. This decision point’s dataset has 9976 rows, four features and two possible outcomes. The features are Observations, Entity Type, Entity ID and Amount. It is a
binary decision point with the outcomes Debit Another and Impose Debit.

5.3.1 Feature Analysis

The first feature is Observations which is an Open Text Feature, therefore, this feature has a lot of different possible values in relation to the number of observations. However, in order to reduce the dimensionality, we extracted some patterns from the features. For example, the string ”Debit Authorized by Insurer” was present in a lot of observations and therefore we extracted this patterns while reducing the noise.

This decision point’s second feature is Entity Type, which is a Discrete Textual Feature with six possible values. Their distribution is in Figure 5.10, where we can see that there is a skewed distribution towards the value FORN.

The third feature is called Entity ID, which also is a Discrete Textual Feature. Given the name of this feature, one could guess that it is related to the previous feature, Entity Type, and has explained before, the Naive Bayes algorithm depends on the features being independent. Therefore, we had to assess if the features’ values had any correlation. This independence is hard to check since we don’t know much about the features besides their names and their values. One way to check is to see if for each value in Entity Type there is a one-to-one mapping in Entity ID. And since Entity ID has 311 possible values versus the six possible values in Entity Type, one can assert the the features are independent. Also, in this feature, there were a lot of ”Nan” values, which were dropped. The most common values distribution can be analyzed in Figure 5.11.
This decision point’s fourth and last feature is **Amount**, a **Numerical Feature**. Just like in the previous decision point’s numerical feature, the missing values were filled in with the mean. We filled it in instead of deleting these points because that would result in a high loss of information, since there was a high number of observations with missing values. The feature can be visualized in Figure 5.12, both in a boxplot and a distribution plot. This feature’s mean is 77, with a 25, 50 and 75 percentiles of 23.72, 32.79 and 67.55 respectively. In the distribution plot we can see that most values are between 0 and 100, with values going up to 2000.

### 5.3.2 Class Analysis

Decision point **Responsibility Validation** is a binary classification point with the possible outcomes being **Debit Another** or **Impose Debit**. The class distribution can be analyzed in Figure 5.13, where we can see that the majority Class is **Debit Another**, represented in 66% of the observations.
5.4 Communicate DoA Advise

Communicate DoA Advise is the last decision point and has the smallest footprint of all decision points. It only has 301 observations, one feature, called Rejection Description and two possible outcomes, Accept or Reject.

5.4.1 Feature Analysis

The one and only feature for this decision point is called Rejection Description and it is an Open Text Feature with a high degree of variability, and with no observable patterns to be extracted.

5.4.2 Class Analysis

Communicate DoA Advise is a binary decision point with two possible outcomes, Accept or Reject. As can be observed in Figure 5.14, there is a high class imbalance, with 83% of the observations with this outcome. However, since there aren’t many observations to work with, we didn’t perform any sampling techniques in order to balance the dataset.
Figure 5.14: Communicate DoA Advise Class Distribution
Chapter 6

Classifiers Results

6.1 Evaluation Metrics and Methods

To evaluate the classifiers we have to assert how well it predicts new observations (observations where the outcome is unknown). To do so, we carried out an automatic classifier evaluation, where the classifier for each decision point is evaluated in relation to some metrics.

A classifier is an algorithm or mathematical function that maps input data to a category or class. In this solution, each decision point will be turned into a classification problem. There are several types of classifier algorithms (e.g. Decision Trees, Naïve Bayes, Support Vector Machines, etc) that can be divided into sub-classes. Machine Learning divides classification onto binary, multi-class, multi-label and hierarchical tasks. Each one of this classes has its own methods of evaluation, therefore, to choose an evaluation method we first need to determine in which of this classes our classifier fits the best.

Each activity might have one or more outcomes, however, the output of the classifier must consist of only one out of these outcomes. The outcomes will the the classes that our classifier will predict. If an activity only has one possible outcome, it won’t be considered for the classifier. If an activity has 2 possible outcomes, we will use a binary classifier. At last, if an activity has more than two possible outcomes, we will use a multi-class classifier.

As explained in the previous section, the output of the system (using Naïve Bayes) will consist of one or more decisions with its associated probability. One could ask, isn’t it a multi-label classification then? Simple answer is yes but no. We are in fact predicting more than one label, however, in the data there is only one class per instance out of 2 or more possible classes. In machine learning terms, this is in fact a binary or multi-class classification problem and not a multi-label one. And for automatic evaluation practices we have to consider this as such. Therefore, to evaluate the quality of the Naïve Bayes classification, only the highest probability decision will be considered.

In addition to the evaluation of the Naïve Bayes classifier, the Decision Tree classifier and the Baseline solution will also be evaluated and the results will be compared.

First, let’s focus on the evaluation of the binary classifiers. Consider a decision point where two outcomes are possible, Outcome1 and Outcome2. It’s a common practice to distinguish between two output classes in a binary classifier as Class1 and ¬Class1. So let’s call our outcomes Outcome1 and ¬Outcome1. Thus, an input in the
training data that "decides" Outcome1 is considered as positive and negative if it "decides" ¬Outcome1. We can then consider an input as Classified positive if the classifier maps it to Outcome1 and as Classified negative if the classifier maps it to ¬Outcome1. The behavior of the classifier can then be represented in a matrix, commonly called confusion matrix, represented in table 6.1.

<table>
<thead>
<tr>
<th>Classified positive</th>
<th>Classified negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>True positive</td>
</tr>
<tr>
<td>Negative</td>
<td>False positive</td>
</tr>
</tbody>
</table>

Table 6.1: Binary Confusion Matrix

An input is a true positive (tp) if it is positive in the test data and is classified positive by the classifier (i.e. right classification). An input is a true negative (tn) if it is negative in the test data and is classified positive by the classifier (i.e. right classification). An input is a false positive (fp) if it is negative in the test data and is classified positive by the classifier (i.e. wrong classification). An input is a false negative (fn) if it is positive in the test data and is classified negative by the classifier (i.e. wrong classification).

To fill in this matrix our classifier has to be trained and tested. To do so a technique called k-fold cross-validation will be used, cf. [22]. Simply put, to use this technique, the data belonging to a decision point under test is split into k parts. k − 1 are used to train the classifier and 1 part is used to test. Then the parts are re-shuffled and the process is repeated.

Each binary classifier will have one associated confusion matrix. After this process is finished, we estimate the classifier quality with 4 different measures, as defined by the authors in [6]:

- **Accuracy**: Overall effectiveness of a classifier
  \[
  \frac{tp + tn}{tp + fn + fp + tn}
  \]

- **Precision**: the number of correctly classified positive examples divided by the number of examples labeled by the system as positive
  \[
  \frac{tp}{tp + fp}
  \]

- **Recall (sensitivity)**: the number of correctly classified positive examples divided by the number of positive examples in the data
  \[
  \frac{tp}{tp + fn}
  \]

- **Fscore**: Relations between data’s positive labels and those given by a classifier. Consists of a combination of the previous two measures.
  \[
  2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
  \]

So, for each binary decision point this evaluation method will be carried out and these measures will be calculated to estimate its overall quality.

Let’s now focus on the evaluation of multi-class classifiers. The k-fold cross-validation will be used as well. However, since the output is not binary, the output of the tests will be different and the confusion matrix has to be
adapted to account for the multiple classes. Consider an activity with \( n \) different possible outcomes, therefore we can represent the different outcomes as Out1, Out2, ... Out\( n \). If an input belongs to the class \( i \) in the training data, with \( 1 \leq i \leq n \), it will appear in the Out\( i \) row of the confusion matrix. If an input is classified as \( i \) by the classifier it will appear in the Classified Out\( i \) column in the confusion matrix. Therefore, the confusion matrix will hold more information for each decision point and is represented in table 6.2

<table>
<thead>
<tr>
<th>Classified Out1</th>
<th>...</th>
<th>Classified Outi</th>
<th>...</th>
<th>Classified Out( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out1</td>
<td>122</td>
<td>...</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Out( i )</td>
<td>3</td>
<td>...</td>
<td>213</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Out( n )</td>
<td>22</td>
<td>...</td>
<td>7</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.2: Multi-class Confusion Matrix

The correct classifications will be on the main diagonal of this matrix. The true positives for class \( i \) is given by the count in the entry \((i, i)\). The true negatives is given by the sum of the counts in the main diagonal, excluding the entry \((i, i)\). The false positives are given by the sum of the counts in the \( i \)th column. The false negatives are given by the sum of the counts in the \( i \)th row.

After this counts are determined, the quality of the classifier is calculated with the following metrics, as defined by the authors in cf. [6]:

- **Average Accuracy**: The average per-class effectiveness of a classifier.
  \[
  \sum_{i=1}^{n} \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i} \times \frac{n}{n}
  \]

- **Precision (with macro-averaging)**: An average per-class agreement of the data class labels with those of a classifiers.
  \[
  \sum_{i=1}^{n} \frac{tp_i}{tp_i + fp_i} \times \frac{n}{n}
  \]

- **Recall (with macro-averaging)**: An average per-class effectiveness of a classifier to identify class labels
  \[
  \sum_{i=1}^{n} \frac{tp_i}{tp_i + fn_i} \times \frac{n}{n}
  \]

- **Fscore**: Relations between data’s positive labels and those given by a classifier based on a per-class average.
  \[
  2 \times \frac{precision \times recall}{precision + recall}
  \]

Each decision point has a set of associated measures that expresses its quality. The overall quality of the system, regarding the classifiers, might be given by all the measures or by a mean of all measures, i.e. the overall accuracy is given by the mean of all accuracies, and similarly for all the other measures.
6.2 Evaluation Results

In this section, we will present the detailed results achieved by the classifiers in four decision points, the ones presented in Chapter 5. We evaluated every classifier (Baseline, Decision Trees and Hybrid Naïve Bayes) in every decision point in order to determine if the developed and proposed approach is the best suitable one. All the classifiers were trained and evaluated with the same datasets and with the same training and test splits.

In the detailed evaluation we will focus on the classifiers metrics explained in Section 6.1 while also showing the confusion matrix for the Hybrid Naïve Bayes.

After presenting the detailed evaluation for the chosen decision points, we will also present a comparison between all the classifiers but regarding all the decision points, to get a better understanding of how the results compare.

However, for decision point Passage to Budget Analysis, it was not possible to train and evaluate the Decision Tree classifier, since this classifier’s training phase never ended, and therefore we couldn’t evaluate it.

6.2.1 Notify Repair State

We trained the three classifiers with this decision point’s dataset and achieved the results in Figure 6.1. As we can see, the Hybrid Naïve Bayes classifier outperformed the other two, especially in the precision, recall and fscore measures.

The high accuracy for the Baseline classifier is easy to understand, since there is such a high bias towards a class, the Baseline, which is only going to pick the majority class, gets the prediction right the majority of the times. However, since the other measures (precision, recall and fscore) don’t only focus on the times that the classifier was right, these one were much lower than the accuracy.

The Hybrid Naïve Bayes results, although they might be considered low, (results in the 50%-70% range aren’t usually considered to be good), it was still capable of performing better than the other classifiers, proving to be a better choice. The low results, however, might be explained by the high number of classes versus the low number of features to learn from. In the confusion matrix in Figure 6.2, we can observe that most of the times, the classifier gets the predictions right (main diagonal). The majority of the times the classifier is wrong, is when it predicts the class Repaired, which is the majority class, but the instance doesn’t actually belong to that class. We believe that with more features and more observations, this decision point would have much better results.
6.2.2 Debit Notification Validation

The final results for the Debit Notification Validation are shown in Figure 6.3, where we can see the comparison between the three classifiers and the Hybrid Naïve Bayes classifier outperforming the other two, however, by a small margin in comparison to the Decision Tree classifier.

In comparison to the other decision points, Debit Notification Validation was the decision point with the best results in all measures. This might be due to the fact that it has the most features, has a set of features that are different, with a feature of each type, is somewhat balanced (if we had more observations we could have considered balancing the dataset) and is a binary classifier, which causes less uncertainty.

Taking a closer look at the confusion matrix in Figure 6.4, we can also see that most of the predictions are in the main diagonal, an therefore are correct. Also most of the instances where the classifier is wrong, is when the
true class is **Accept Debit**, but the classifier predicts **Reject Debit** and this might be due to the higher number of **Reject Debit** observations, which can skew the results of the classifier in that outcome’s direction.

![Classifier Comparison](image1)

Figure 6.3: Classifier Comparison for Decision Debit Notification Validation

![Confusion Matrix](image2)

Figure 6.4: Hybrid Naïve Bayes Confusion Matrix for Decision Debit Notification Validation

### 6.2.3 Responsibility Validation

Now we are going to show and discuss the results for decision point **Responsibility Validation**. As we can see in Figure 6.5, similarly to the previous decision point, the **Hybrid Naïve Bayes** was again the best performing algorithm, however with only a small advantage in relation to the **Decision Trees** Classifier.

This decision point’s footprint was similar to the previous one’s, in the sense that it is a binary decision point with four (five in the previous), with different features, with patterns extracted from the **Open Text Features** and with a similar size footprint. This decision point’s classifier was the second best in the evaluation faze, which
shows a pattern in this subject. The classifiers are able to predict and their implementation is sound in some cases, with results getting better with the quality of the dataset itself.

Taking a closer look at the confusion matrix in Figure 6.6, we can see that most predictions are in the main diagonal. However, most of the errors happened when the classifier predicted Impose Debit but the true correct decision was Debit Another.

![Classifier Comparison for Decision Responsibility Validation](image)

**Figure 6.5: Classifier Comparison for Decision Responsibility Validation**

![Hybrid Naïve Bayes Confusion Matrix for Decision Responsibility Validation](image)

**Figure 6.6: Hybrid Naïve Bayes Confusion Matrix for Decision Responsibility Validation**

### 6.2.4 Communicate DoA Advise

Now moving on to the last decision point, **Communicate DoA Advise**, we can see by the graph in Figure 6.7 that the results in this decision point were subpar, especially when compared with the results achieved by the Decision Tree algorithm. As we saw in Chapter 5.4, this decision point’s dataset was lacking features and observations, which in turn resulted in a a wrongfully trained Hybrid Naïve Bayes classifier.
This results, however, allow us to draw conclusions in regards to the applicability of the Hybrid Naïve Bayes, since we can argue that in these cases where the dataset is not as good, we should use a different classifier, like the Decision Trees algorithm.

However, from our experience with the dataset, this situation is the minority, and therefore we can argue that the approach suggested in this dissertation, can and should be used in order to achieve better results.

Taking a closer look at the confusion matrix produced by the Hybrid Naïve Bayes algorithm, we can see that the classifier is very good at predicting the Accept cases, where it actually got all the predictions correct, which explains the high precision measure in Figure 6.7.

![Figure 6.7: Classifier Comparison for decision point Communicate DoA Advise](image)

![Figure 6.8: Hybrid Naïve Bayes Confusion Matrix for Decision point Communicate DoA Advise](image)
6.2.5 Overall Comparison

In this section we are going to present the overall comparison between the classifiers, regarding all the decision point. We calculated these results by evaluating all the decision points with the three classifiers, gathering the metrics by decision point and by classifier and calculating the mean of those measures. For example, for the Hybrid Naïve Bayes classifier and for the accuracy metric, we gathered the set \(\text{accuracies} = \{\text{dec1acc}, \text{dec2acc}, ...\}\), where \(\text{dec1acc}\) and \(\text{dec2acc}\) are the accuracy measures for given decision points 1 and 2, and then calculated the mean accuracy for the Hybrid Naïve Bayes classifier with \(\frac{\text{sum(accuracies)}}{\text{length(accuracies)}}\). We repeated this procedure for all the classifiers, for all the metrics and crunched the results displayed in Figure 6.9.

We can observe that the Hybrid Naïve Bayes was the classifier with the best overall results, outperforming the Decision Trees approach in every decision point except in the Communicate DoA advise. Not only did it achieve better results in most of the decision points, but the advantages are also relevant, especially on the precision and recall measures (and subsequently on the fscore measure), making it a good alternative to the algorithms used in similar solutions.

![Classifier Comparison](Image)

Figure 6.9: Classifier Comparison between all the classifiers
Chapter 7

Conclusions

Business processes create a trail of execution, showing how the instances were carried out. In many cases, these data are not being used. Process Mining and Decision Mining are valuable tools for business process managers, that make use of the valuable event logs. Tools that can help them understand how deployed processes are really carried out. With our solution we can bring the benefits of these tools to the front-line of the business processes, providing real-time operational support. With this approach, our system promotes a cooperation between the business process actors and the recommendation system.

We presented the state of the art of these topics and our solution, that will follow many of the concepts and methodologies proposed by other authors and propose others that have not yet been researched by the community but showed promising results, described in this paper. Therefore, we hope to enrich this field of study with novel ideas.

Machine Learning methods and algorithms, together with the analysis of the event logs, can create interesting solutions, able to provide intelligent recommendation systems. In this document we introduced our approach which can be applied in many other environments where process payload and decision data are available.

Most solutions in this area of Decision Mining and Recommendation Systems use Decision Trees as the main classifier algorithm or just provide recommendations that don’t take full advantage of what is possible to learn from the event logs. Decision Tree classifying is a deterministic algorithm, and in our solution, we implemented a stochastic approach focused on the control-flow and data-flow perspectives of the process. The Hybrid Naïve Bayes Classifier proposed in this paper takes full advantage of the historic event logs to learn from past executions, guide running cases and learn from new observations, turning the process into an adaptive and cooperative procedure. The results, evaluated by the defined methods, were better than those of presented in similar solutions.

Even though, the results were marginally better, in terms of automatic classifier evaluation, we believe that our approaches relating to the use of a probabilistic algorithm, the decision point identification and how the datasets are evaluated, can provide relevant information and bring benefits to the execution of business processes.
7.1 Achievements

In this thesis project we developed a solution in the field of Process Mining, and in particular, in the field of Decision Mining applied to Process Mining, that follows a methodology which hadn’t yet been explored and applied.

We had the objectives of providing a solid recommendation system, capable of providing real-time recommendations and ranking them in terms of their fitness to the situation.

With the solution proposed on this document and with the results obtained, we achieved the objectives we had set and believe that this approached can enrich the field with new ideas and concepts.

7.2 Future Work

A few features and improvements were left for future work since they fell out of the scope or were found to be hard to implement given the state of the project assets. In this section we are going to explain what was left out and why. Most of the problems were due to the quality of the event logs. Given its raw state it was hard to work with.

In the dissertation proposal, we proposed to develop a performance predictor that would work together with the recommendation system in order to provide an insightful performance prediction associated with the decision recommendation. To do so, we analyzed the process instance time measures of the event logs, for example, plotting the time against the number of activities to see if there was any correlation, since in the proposal we proposed to predict the time measure with the number of expected activities as prediction measure. However, plotting the data showed that there was no pattern in the data and process instances with 3 activities often took the same time to finish as process instances with 40 activities. Then we thought of using the process payload to predict the time. However, the process instances had different payloads, making it hard to select a set of features to use (just like the decision points we didn’t use because of lack of features). One possible cause for the time variance might be due to the fact that the process instances or the activities aren’t closed in the system after they are finished, resulting in activities with inflated time measures which do not reflect the truth and therefore are not sufficient or appropriate to apply a regression algorithm.

As explained in Chapter 1.3, the event logs and, in particular, the decisions made by the business process actors are not validated and therefore the might have errors. In a future iteration, the decisions could be validated by someone with business knowledge, which might lead to better results by the classifiers.

In the future, this solution can be integrated with the deployed process and provide a recommendation system as a web service. The results show that the classification process is very fast and the classifiers predict with a high degree of certainty in some decision points, while others can be optimized in order to reach a better state and be provided as a service.

7.3 Communication

During the writing of this document and the development of our solution, we also proposed papers to three different conferences on the topic, they were:
We were accepted in the CAPSI2018, a Portuguese conference hosted by the Portuguese Information System Association, with a paper regarding this solution and its results. The presentation in the conference was already carried out, where we received a good feedback in regards to the solution and the our presentation.

The papers were not accepted in the other conferences, with the best scores being Weak Accepts by the reviewers of Coopis, a rank A conference, in regards to their recommendations for the paper approval, originality and relevance of the solution.

---

2 http://www.otmconferences.org/index.php/conferences/coopis18
3 https://www.edoc2018.conf.kth.se/
Bibliography


