Real-Time Business Process Recommendations

João Miguel Valentim Rodrigues - 78672
Supervisor: Professor Pedro Sousa
Supervisor: Engenheiro José Rodrigues

Instituto Superior Técnico - Universidade de Lisboa

***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.**

***Index Terms*—Business Process Mining; Process Mining; Decision Mining; Machine Learning; Event Logs**

I. 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 hands. 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.

A. 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.

B. Document Outline

In this document, first we are presenting similar solutions in Section II, followed by the case study we worked with in Section III. In Section IV we thoroughly explain the solution, before demonstrating its results in Section V. Finally, we end this document with the conclusion in Section VI.

II. 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 Process Aware Information Systems (PAIS). In this section, a literature review on the subject is introduced, mainly focusing on **Decision Mining** and **Recommendation Systems**.

A. Process 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.

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.

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], [8], [9], [10], [11]. In this book, a plug-in for the tool **ProM**[1] 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.

B. 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

[1] http://www.processmining.org/prom/start

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 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.

Table I
CONTROL PERSPECTIVE

Case ID	Activity ID	Task Number	Outcome
213111	Analyze Budget	213	Approve
213111	Deliver Device	214	Delivered
123400	Analyze Budget	401	Reject

Table II
DATA PERSPECTIVE

Task Number	Data Object ID	Data Object Value
213	Amount	120.00€
213	Own Brand	Yes
401	Amount	80€
401	Own Brand	No

III. CASE STUDY

The solution follows a methodology that can be applied in multiple environments. However, in the scope of this project, 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 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 has experienced some changes since then. Therefore, we are working with the data from 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 and the real-time event logs will be provided by a Oracle™BPM Engine.

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 I and II.

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 activity's 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 executed that activity.

IV. 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.

A. 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 a activity might be the succeeding activities that are executed, as is presented in [1].

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 III 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 III
Example of an activity with two different outcomes

CaseID	Activity Name	Outcome
537132415	Analyze Budget	Accept
568075942	Analyze Budget	Reject

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.

B. 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 and 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 think that these are the 3 ways that should be considered and used accordingly in other solutions in this area.

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, 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 features;

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 IV.

1) Results -: Each decision point had between 100 and 115 features. Various types of features were identified. These features are relative to the data-set 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;
- 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 (dimensionality reduction).

The number of features had to be reduced. 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 IV) 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.

What was interesting to notice while selecting the features is that no business knowledge was needed. Following these techniques, there is no need to know more about the process and good results can be achieved.

C. 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 Section II;
- **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.

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

Table IV
FOOTPRINT TABLE EXCERPT FOR DECISION POINT DEBIT NOTIFICATION VALIDATION

Constestation Reason	roleENT	Amount	Observations	Cause	Outcome
No Schedule Violation	SCR	41.56	Debit Authorized by Insurer	Exchange Authorized	Reject Debit
NA	FORN_6008	23.73	Client in Store	Exchange Authorized	Accept Debit

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.

The 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:

$$P(value1|o1) = \frac{c(value1) + 1}{c(o1) + V}$$

As for continuous features, we approximated the possible values to a **Gaussian distribution**, calculating the μ (mean) and σ (standard deviation).

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 1 and 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 and training the chosen classifier).

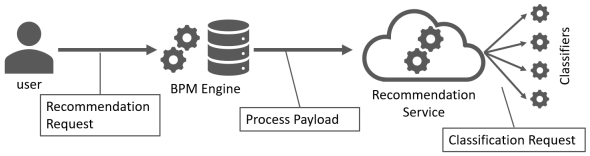

Figure 1. Recommendation Request

To exemplify what the Classification Request and the Recommendation in Figures 1 and 2 look like, consider the classifier trained with the data in Table IV. A Classification Request looks like a row in Table IV 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 V, 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.

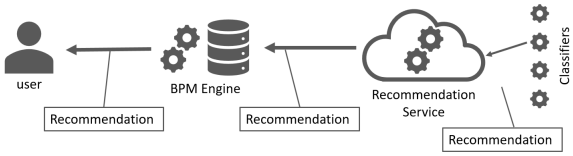

Figure 2. System Response

Table V
RECOMMENDATION EXAMPLE

Outcome	Probability
Accept Debit	23%
Reject Debit	77%

D. Architecture

In this section the project's Architecture is presented in figure 3. In this architecture, presented in a "Pipes and Filters" UML[2] Component Diagram 3, the training phase of the solution is specified and each component is described.

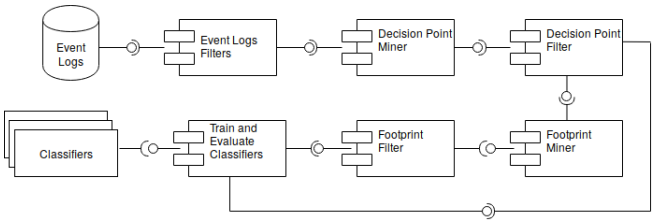

Figure 3. Solution Architecture

1) Event Logs: The training phase starts with the raw historic Event Logs, provided by the BPM engine. The Event Logs come in csv[3] form and their general structure is presented in Section III.

[2]https://www.omg.org/spec/UML/About-UML/
[3]https://en.wikipedia.org/wiki/Comma-separated_values

2) Event Logs Filters: First, the Event Logs need to be carefully analyzed and filtered. As explained in III, 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**.

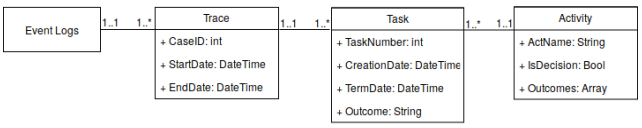

Figure 4. UML Class Diagram

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

3) Decision Point Miner: The decision point identification is an important step and is thoroughly explained in Section IV-A. This component runs through the set of activities flagging as **decision points** the activities that have 2 or more **outcomes**.

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.

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 IV-B 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.

6) Footprint Filter: After finding the footprints (one for each decision point), they were carefully analyzed, performing an Exploratory Data Analysis (EDA), 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.

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 Section V. 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).

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.

V. Evaluation and Results

A. Decision Points Evaluation and Results

We will now talk about the discovered decision points with the approach presented in Section **??**. 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;

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 5, 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 Section II, 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.

B. Classifier Evaluation and Results

To evaluate the classifiers, an automatic evaluation was conducted. To carry this evaluation, a data-set and a classifier are needed. The data-sets are the footprints (generated by our methods) and the classifiers are the ones that were presented in the previous section. To explain how the evaluation was carried out, we will focus in only one decision point with the Hybrid Naïve Bayes classifier to present a more detailed evaluation and then present the overall evaluation in comparison with the other classifiers.

The metrics chosen were the usual Accuracy, Precision, Recall and Fscore with macro averaging for multi-class classifiers

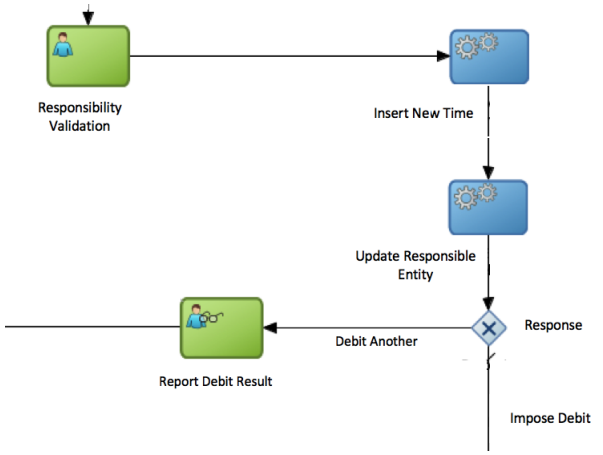

Figure 5. BPMN model regarding the activity Responsibility Validation.

(decision points with more than two possible outcomes), as presented in [6]. To carry out the automatic evaluation, we performed a 80%/20% train/test split.

One important thing to note is that the case study's event logs were created by the deployed process's BPM Engine, and were not subject to any human verification. Meaning that the footprint tables express the decisions took by the process actors and therefore may contain errors, if the actors themselves didn't make the right decisions, since it wasn't subject to any further verification. However, we made the assumption that, in their majority, the decisions are correct and the defective observations are not enough to jeopardize the learning and classification process.

C. Detailed Evaluation -

The decision point which we will focus on is called Debit Notification Validation. It has 11929 rows (observations) and 5 features, as can be observed in Table IV, and are further explained:

1) **Contestation Reason** is a discrete textual feature with 3 possible values a a high bias towards the value "No Schedule Violation", appearing 91% of the times;
2) **roleENT** which is also a discrete textual feature with 146 possible values;
3) **Amount** which is the only numerical feature with a mean of 81. The missing values were filled in with the mean;
4) **Observations** is an open text feature where a few patterns were found and extracted. For example, one of the patterns was the sentence "Debit Authorized by Insurer" which could appear alone or not in the sentence, usually with information about the name of the process actor who executed the action activity. The extra information was removed in order to reduce the dimensionality of this feature;
5) **Cause** is also an open text feature where patterns like the sentence "Exchange Authorized" or the word "Redline" followed by some ID's could be observed. Therefore we extracted these patterns to reduce the dimensionality;

Table VI
CONFUSION MATRIX FOR HYBRID NAÏVE BAYES CLASSIFIER IN DECISION POINT DEBIT NOTIFICATION VALIDATION

	Predicted Accept Debit	**Predicted Reject Debit**
Actual Accept Debit	500	24
Actual Reject Debit	5	1857

These were the chosen features, from 112 starting features, applying the methods presented in Section IV-B and were the ones that produces the best results.

This decision point has two possible outcomes, making it a binary classification problem. The possible outcomes are "Accept Debit" or "Reject Debit" with a 22%/78% split respectively. Therefore, some class imbalance can be observed. However, this didn't jeopardize the results, therefore there was no need to perform a class balancing. In other decision points, however, this had to be done, by excluding a percentage of the main class, in the training set.

The evaluation results will now be presented. First we present the confusion matrix, in Table VI, where one can see the number of correctly predicted labels in the main diagonal of the matrix.

As can be observed, most of the predictions by the classifier are correct. From these numbers we can derive the metrics explained before. The derived metrics are displayed in Figure 6, and we can see that the results are close to 99% across all metrics. Therefore, we can derive that the classifier predicts the right decisions in the great majority of the times.

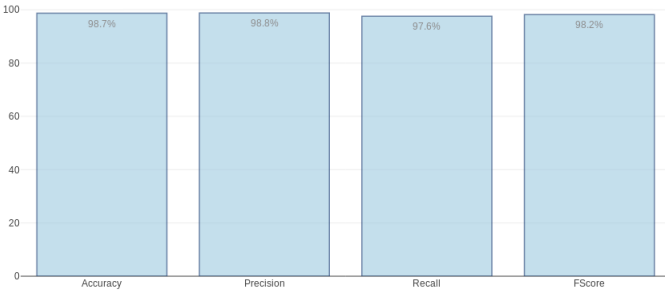

Figure 6. Evaluation Metrics for Hybrid Naïve Bayes Classifier in decision point Debit Notification Validation

1) Overall Evaluation -: In order to validate our solution we also evaluated the other developed classifiers, with special attention to the Decision Trees Classifier, since it is the one used in most of the solutions in this area. To conduct this overall evaluation we trained every classifier with every decision point, calculated the same metrics and compared it with one another.

Overall the Hybrid Naïve Bayes Classifier was better across all metrics, however the Decision Trees Classifier did show

some better results in two decision points, but just in the **accuracy** metric. To show the overall comparison between the classifiers, we calculated the mean scores for all the decision points, by metric. The results are displayed in Figure 7

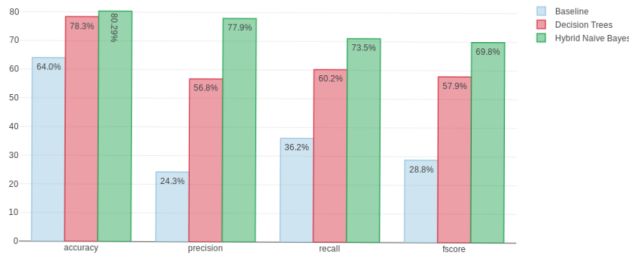

Figure 7. Evaluation Metrics comparison between all classifiers in all decision points

VI. Conclusion

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.

References

[1] Process mining: conformance and extension - AA Rozinat - 2010

[2] Van der Aalst, Wil MP. "Extracting event data from databases to unleash process mining." BPM-Driving innovation in a digital world. Springer, Cham, 2015. 105-128.

[3] Van Der Aalst, Wil, et al. "Process mining manifesto." International Conference on Business Process Management. Springer, Berlin, Heidelberg, 2011.

[4] Guyon, Isabelle, and André Elisseeff. "An introduction to variable and feature selection." Journal of machine learning research 3.Mar (2003): 1157-1182.

[5] Van der Aalst, Wil MP. "Extracting event data from databases to unleash process mining." BPM-Driving innovation in a digital world. Springer, Cham, 2015. 105-128.

[6] Sokolova, Marina, and Guy Lapalme. "A systematic analysis of performance measures for classification tasks." Information Processing & Management 45.4 (2009): 427-437.

[7] Ghattas, Johny, Pnina Soffer, and Mor Peleg. "Improving business process decision making based on past experience." Decision Support Systems 59 (2014): 93-107.

[8] Rozinat, Anne, and Willibrordus Martinus Pancratius Aalst. Decision mining in business processes. Beta, Research School for Operations Management and Logistics, 2006.

[9] De Leoni, Massimiliano, and Wil MP van der Aalst. "Data-aware process mining: discovering decisions in processes using alignments." Proceedings of the 28th annual ACM symposium on applied computing. ACM, 2013.

[10] Wetzstein, Branimir, et al. "Monitoring and analyzing influential factors of business process performance." Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International. IEEE, 2009.

[11] De Leoni, Massimiliano, Marlon Dumas, and Luciano García-Bañuelos. "Discovering branching conditions from business process execution logs." International Conference on Fundamental Approaches to Software Engineering. Springer, Berlin, Heidelberg, 2013.

[12] Schonenberg, Helen, et al. "Supporting flexible processes through recommendations based on history." International Conference on Business Process Management. Springer, Berlin, Heidelberg, 2008.

[13] Motahari-Nezhad, Hamid Reza, and Claudio Bartolini. "Next best step and expert recommendation for collaborative processes in it service management." International Conference on Business Process Management. Springer, Berlin, Heidelberg, 2011.

[14] Van der Aalst, Wil MP, Maja Pesic, and Minseok Song. "Beyond process mining: from the past to present and future." International Conference on Advanced Information Systems Engineering. Springer, Berlin, Heidelberg, 2010.