TECHNICAL UNIVERSITY OF MUNICH

TUM Data Innovation Lab

Machine Learning powered Process Mining: Data Analysis and Prediction

Authors:	Margarita Ageeva, Ahmed Ayadi, Sebastian Roßner, Olha Tupko, Keesiu Wong
Supervisor:	Jan Philipp Thomsen, Jerome Geyer-Klingeberg
Advisor:	Prof. Dr. Massimo Fornasier, Dr. Ricardo Acevedo Cabra, Juliane Sigl
Date:	February 17, 2018

Contents

List of Figures

List of Tables

Acronyms

BPMN	Business Process Model and Notation
ERP	Enterprise Resource Planing
KPI	Key Performance Indicator
MAE	Mean Absolute Error
PI	Process Intelligence
PO	Purchase Order
P2P	Purchase-to-Pay

Abstract

This report is a summary of our work within the TUM Data Innovation Lab project entitled "Machine Learning powered Process Mining". The project was proposed by Celonis SE, the market leader in Process Mining technologies, and consisted of two major parts: the *Data Analysis* part and the *Machine Learning* part.

The *Data Analysis* part was split into two sub-phases. The first was dedicated to the creation of Process Mining analyses for Microsoft Dynamics AX. This resulted in more than ten analyses currently deployed and being used by Celonis' customers from the AX world, thus enlarging Celonis' market possibilities. In the second sub-phase, we came up with three new analyses widely applicable across different processes and ERP systems. These analyses will allow Celonis' customers to objectively measure the complexity of their processes, thoroughly analyze their throughput times, and precisely investigate the undesired activities therein.

The *Machine Learning* part we provide an end-to-end solution that allow customers to predict. The problem was a classification task, and numerous methods were investigated and compared for this purpose. At the end, we decided for a solution achieving around 90% f_1.

Through the extension of the Celonis software to Microsoft AX on the one hand, and the innovative combination of Process Mining and Machine Learning on the other hand, this project has been successful.

This version is a shorted version from the original. Company confidential information has been removed.

1 Introduction

In this chapter, we give a brief introduction into Process Mining, the discipline that is at the very core of both Celonis in general and this project in particular. Thereby, we will also introduce the reader to the notation we will be using throughout this report. Finally, we give a broad overview about the project and the further chapters.

1.1 Process Mining

According to Prof. Wil van der Aalst, considered as one of the *Godfathers of Process Mining*,

> “ Process Mining is a relatively young research discipline that sits between Machine Learning and Data Mining on the one hand and process modeling and analysis on the other hand. The idea is to **discover, monitor and improve real processes by extracting knowledge from Event Logs [...]**” (Aalst 2016, p. 3)

Data Mining and *Process Mining* are in fact similar in that both areas are data-based. Process Mining however is particular in that the data it uses is processual by nature. This is then reflected in the importance of what is called the *Event Log*, a data structure that is central to every Process Mining application. In fact, every Process Mining application starts by a preparation phase where an Event Log is constructed from the logged data of the underlying Enterprise Resource Planing (ERP) system (see subsection 1.1.1). Based on the Event Log and on the tables readily available in such systems, several applications within Process Mining can be identified, perhaps most notably *Process Discovery*, *Conformance Checking* and *Enhancement* (see subsection 1.1.2). This is illustrated in Figure 1.1 where the traditional human-machine interactions are depicted in gray, while the working principle of and the value added by Process Mining are captured in green.

1.1.1 The Event Log

Nearly any kind of today's computer aided systems has a logging functionality for debugging or for the analysis of crashes. To a certain extent, these logged data can be thought of as representing the history of the user interaction with such systems. For example, ERP software (Enterprise Resource Planing) like SAP or Oracle automatically create new entries in their log files whenever a user (e.g. an accountant) interacts with the system, e.g. by changing a certain field in the underlying databases. Thus, by transforming

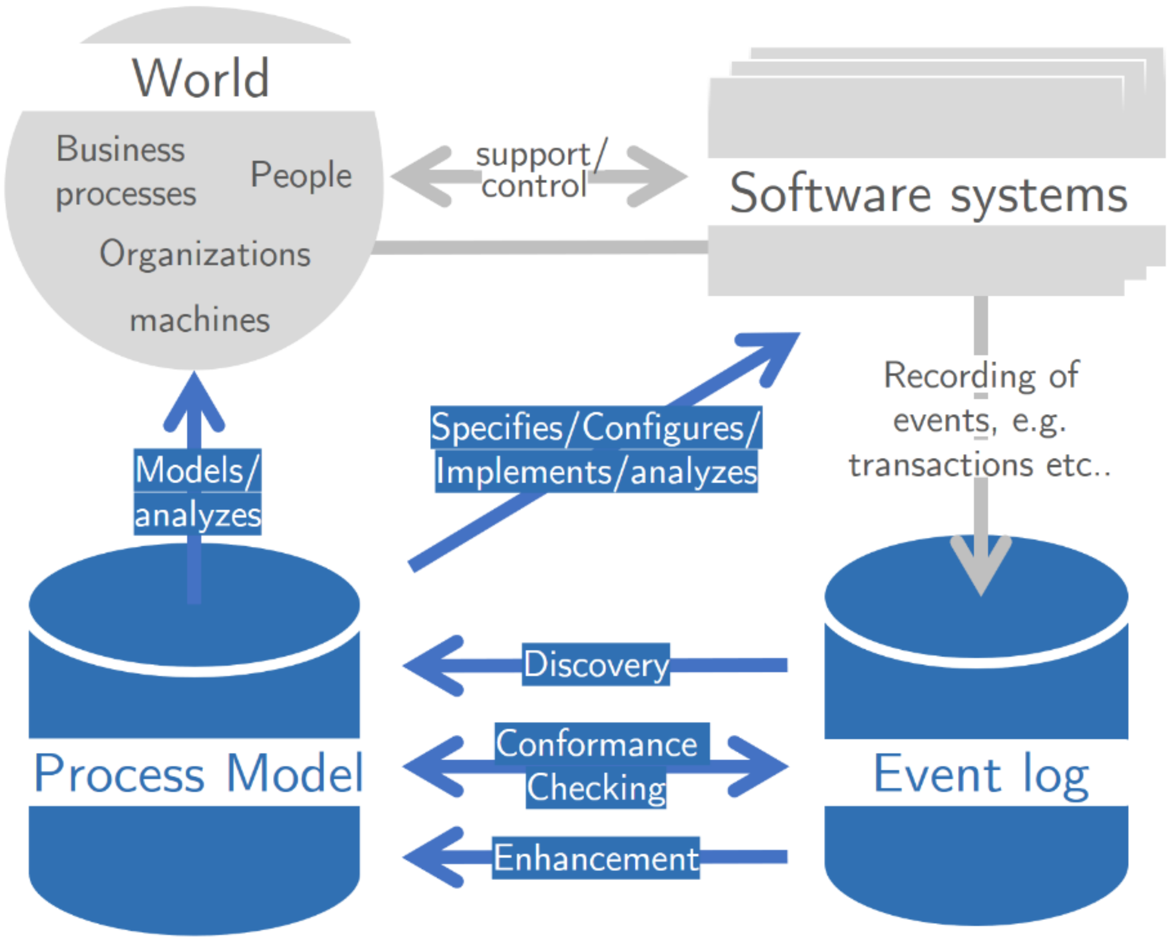


Figure 1.1: An overview about Process Mining (Aalst 2016, p. 32).

these data properly, Process Mining creates value to companies by reconstructing this very history, indeed allowing them to *mine* their processes. For simplicity, assume in the following that a given company X has a certain process, say a Purchase-to-Pay (P2P) process, where in the normal case, customers of X order a given item, receive the item and then pay the invoice within a certain time frame. Now assume that X uses an ERP system that somehow stores all data related to this process. The very first step in Process Mining would then be to transform these data into a human-readable table called the *Event Log*. Conceptually, **this can be thought of as a large table where each row represents a relevant *event* that has occured for a specific *case***. An example of such an Event Log can be seen in Table 4.1. In the following, we give precise definitions of *Events*, *Cases* and the *Event Log*.

Definition 1.1 (Event (Aalst 2016, pp. 130, 131))
An event e is indirectly defined through its *attributes*, e.g., an event may have a time-stamp, correspond to an activity or is executed by a particular person, etc. We denote by $\mathcal{E}$ the set of all possible events, and by AN the set of attribute names. For any event $e \in \mathcal{E}$ and name $n \in AN$, let $\#_n(e)$ be the value of attribute n for event e. In

particular, $\#_{activity}(e)$, $\#_{time}(e)$ and $\#_{resource}(e)$ denote the *activity*, *time-stamp* and *resource* associated with event e, respectively.

In other words, an event is anything that occured in the past and that has been logged by the underlying system. For example, when user A at time t changed the field *price* associated with a certain case, say c_{50} (see definition below), we identify this with an event, say e_4, which among others has the attributes $\#_{activity}(e_4)$ = *change price*, $\#_{time}(e_4) = t$, $\#_{resource}(e_4) = A$, and $\#_{case}(e_4) = c_{50}$.

Definition 1.2 (Trace, Case (Aalst 2016, p.134))
A trace is a unique family of events $\{e_i\} \subset \mathcal{E}$ ordered by their timestamps (i.e. by $\#_{time}(e_i)$). We denote by $\mathcal{T}$ the set of all traces. A case c is a more general construct which, like events, is characterized by several attributes, inter alia a trace $\#_{trace}(c)$, denoted by c^*, such that for all e in c^*, $\#_{case}(e) = c$. We denote by $\mathcal{C}$ the case universe, and by $\mathcal{C}^*$ the set of all traces from $\mathcal{T}$ associated with cases from $\mathcal{C}$, i.e.

$$\mathcal{C}^* = \{tr \in \mathcal{T}, \exists!\, c \in \mathcal{C} : \#_{trace}(c) = tr\}$$

Following up the example above, c^*_{50} would then be the sequence of all events associated with the case c_{50}, including e_4. c_{50} however might have several other attributes, for example the name of the customer involved in the said invoice. Most importantly, each case has an attribute called *id* which allows its unique identification among all other cases. In our example, it holds $\#_{id}(c_{50}) = 50$.

Definition 1.3 (Event Log, (Aalst 2016, p.134))
An Event Log E is a set of traces from $\mathcal{C}^*$. Since each $c^* \in \mathcal{C}^*$ is a sequence of events associated with a certain case $c \in \mathcal{C}$, E can also be seen as a set of events:

$$E = \{\sigma_1, \sigma_2, ..., \sigma_l\} = \{\{e_{1,1}, e_{1,2}, ..., e_{1,N_1}\}, \{e_{2,1}, e_{2,2}, ..., e_{2,N_2}\}, ..., \{e_{l,1}, e_{l,2}, ..., e_{l,N_n}\}\}$$

where $\sigma_i \in \mathcal{C}^*$ and $e_{i,j} \in \mathcal{E}$.

So visually, the Event Log E is a table where each row coincides with one event e, which in turn belongs to the trace $c^* = \#_{trace}(c)$ of a certain case c, hence the name. The combination of the Event Log with other tables that give the other attributes of each case (e.g. $\#_{customer}(c)$ or $\#_{vendor}(c)$ in the Customer and Vendor Master respectively) build the very data structure underlying each Process Mining application. This combination can be easily established by linking these different tables using the unique identifier of each case.

Finally, in some applications like Process Discovery (see subsection 1.1.2), one is merely interested in the sequence of activities that formed past traces in the process (but not in their timestamps for example). This motivates the two following definitions.

Definition 1.4 (Equivalent Traces)
We define the equivalence relation ~ on the set $\mathcal{T}$ as follows:

$$\sigma_1 \sim \sigma_2 \iff |\sigma_1| = |\sigma_2| \wedge \#_{activity}(\sigma_1(i)) = \#_{activity}(\sigma_2(i)) \ \forall i \in \{1, 2, ..., |\sigma_1|\}$$

One can verify that ~ is reflexive, symmetric and transitive. Two traces are thus considered to be ~-equivalent if they involved exactly the same sequence of activities.

Definition 1.5 (Variants)
The set of variants $\mathcal{V}$ is defined as the quotient set of E by ~, i.e. $\mathcal{V} = E/\sim := \{V_1, V_2, ..., V_v\}$, where V_i denotes the i-th equivalence class, and v is the number of such equivalence classes. Thus, we identify each V_i with a variant, i.e. a certain pattern of activites. We further define k_i as the number of traces in E that followed the i-th variant, i.e. $k_i = |V_i|$. Since $\mathcal{V}$ is a partition of E, it holds:

$$|E| = \sum_{i=1}^{v} k_i$$

1.1.2 Process Discovery, Conformance Checking and Enhancement

Now that we have introduced the Event Log, we provide an overview about the three major applications within Process Mining: *Process Discovery*, *Conformance Checking* and *Enhancement*, all of which are fundamentally based on the Event Log.

While Process Discovery consists of generating a process model from an existing Event Log, Conformance Checking uses an existing process model to see how good an Event Log matches it. Process Discovery and Conformance Checking can be combined. For example, ”Conformance Checking can be used to evaluate the performance of a Process Discovery technique” (Aalst 2014, pp. 9, 10).

Enhancement is using this measurement of conformance checking to extend or improve an existing process model. Besides conformance checking there are four competing quality dimensions for the quality of a process model: *fitness, simplicity, precision and generalization* (Aalst 2014, p. 11).

1.2 Project Definition

Celonis is currently considered as the market leader in the Process Mining industry. To consolidate this favorable position, Celonis inter alia relies on **Market Development** by adapting its existing products to more ERP systems and thus enlarging its customer base, and **Product Development** by making use of the rising technological advances in Machine Learning to offer innovative solutions to its existing customers.

This project could be thought of as a further step in the implementation of this strategy. In line with the two strategic axes above, the project was structured in two phases with different strategic foci (see the Ansoff matrix in Figure 1.2).

In the first part, the plan was to start by adapting existing use cases to the Microsoft Dynamics ERP system, thus enabling Celonis to further penetrate a market of mainly middle-sized companies around the globe (Phase 1.1 in Figure 1.2). Thereby, the focus was on the Account Receivables process. The second phase was then to develop new use cases (Phase 1.2 in Figure 1.2). In spite of using the same data as before, this step resulted in three new analyses that are independent of the process under investigation,

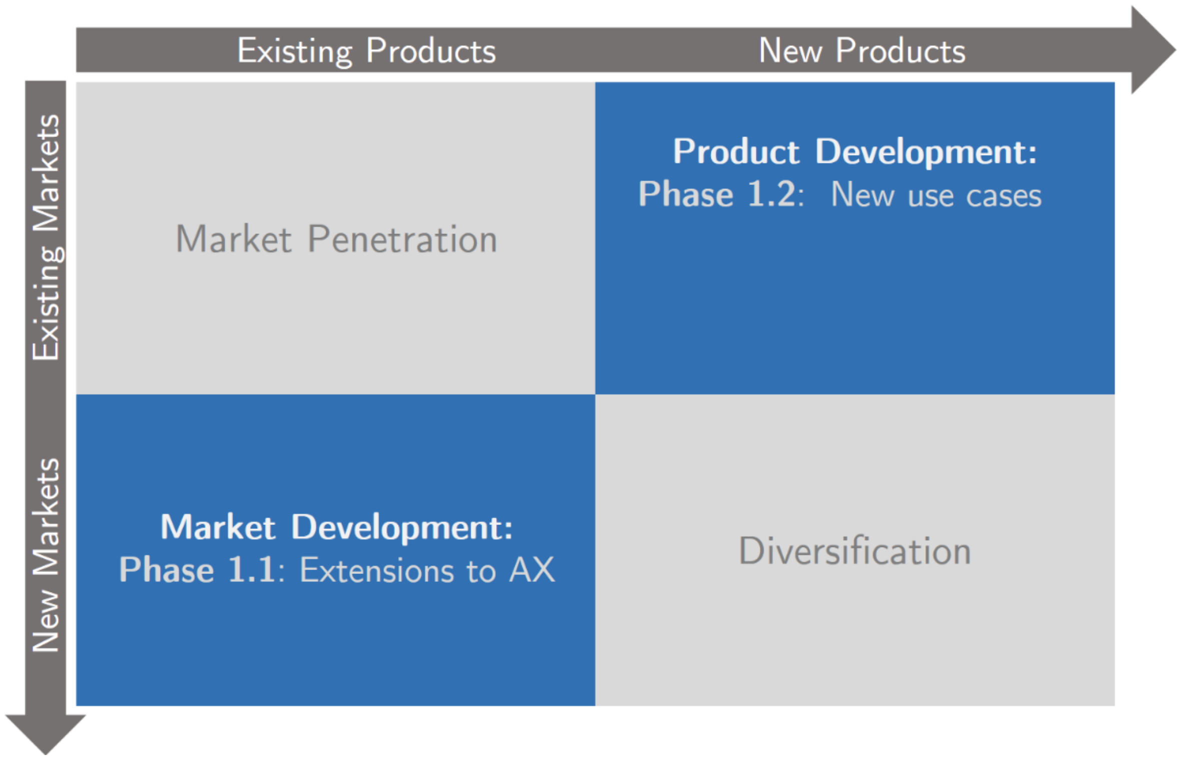


Figure 1.2: Localization of the project within Celonis' strategy

and that can easily be adapted to other ERP systems. A more detailed discussion about the first phase, which in the sequel we will refer to as the **Data Analysis part**, is given in Chapter 2.

The second part focused on empowering the Celonis software with Machine Learning algorithms.

Deeper insights into the second phase, which in the sequel we will refer to as the **Machine Learning part**, is given in Chapter 3.

2 Part I: Data Analysis

The Data Analysis part consisted of two sub-phases. The first sub-phase was dedicated to the creation of Process Mining Analyses for Microsoft Dynamics AX (see section 2.1). The second sub-phase was more innovative and aimed at creating new analyses, that are not yet created by Celonis, but could have significant business decision-making value for its customers. Our focus thereby was on creating analyses that can be applied on any process.

2.1 Existing Use Cases

2.1.1 Overview and Challenges

Focus of this sub-phase was the **Accounts Receivables** process of a Celonis' customer from the automotive industry, which for anonymity reasons will be denoted by X in the sequel. This process usually starts by one of X's customers requiring to purchase one or more of X's offerings (a so-called purchase requisition item), then goes through different steps including but not restricted to creating an invoice, paying the required amount, clearing the invoice and closing the case.

As explained in chapter 1, the data that was made available had first to be transformed into an Event Log linked via a unique identifier ($\#_{id}(.)$) to other relevant tables from the Microsoft database X is using. Thereby, each line in any invoice was defined as being an independent case (in general, invoices include more than one line, i.e. more than one item to purchase). For each such case c, all events related to it, i.e. its trace c^*, were stored in the Event Log, while all other attributes were distributed among the other tables.

This not only required a deep understanding of the meaning of the formulas and the business purposes behind the KPIs included therein, but also of the database architecture Microsoft AX.

Although the Data Analysis Training we took at the beginning of the Project has helped deciphering some of these unclarities. Sometimes, a very simple formula created out of the business requirements transforms into a complex SQL case statement with various conditions when written for Microsoft AX. Also, while working on adjusting existing analyses, we came across several fields or even activities, which were missing from the Microsoft database.

One should understand not only the meaning of specific fields and connections between tables, but also the business purposes of the utilized KPIs. During our work in this phase, we also came up with some suggestions for a more precise calculation of some of the existing KPIs.

2.1.2 Use Cases

This first sub-phase resulted in the following 11 analyses,

- Invoices without sales orders
- Dunning reports
- Cash discounts
- Low-value invoices
- Aging of receivables
- Days sales outstanding
- Dunning blocks
- Process costs
- Overdue invoices
- One-time customers
- Segregation of duties

2.2 New Use Cases

2.2.1 Overview

In this sub-phase, the goal was to find new ideas for use cases demonstrating the power of Process Mining and the capabilities of the Celonis software. Thereby, the ideas had to fulfill two main requirements. First, they had to provide a real added value to the customers by generating new and useful insights into their data. Second, they needed to be as generic as possible by being applicable on any process of any company from any industry. Given our mathematical background, we came up with three new analyses that are rather mathematically oriented than purely business-related: *Process Complexity* (subsection 2.2.2), *Throughput Time Efficiency* (subsection 2.2.3), and *Undesired Activities* (subsection 2.2.4).

2.2.2 Process Complexity

The problem

In the corporate sector, growth is often considered as a primary business objective. In fact, growth is more often than not a necessity rather than a strategic choice. However, when growing, companies inevitably become more complex. Their organizational structures develop layers upon layers and their reporting lines become tangled. As this could come at the expense of performance, managing complexity reveals to be crucial to every organization. In fact, complexity is not necessarily bad: it is rather a two-edged sword, which if managed properly, could create value. To manage complexity though, **companies need to be able to *measure* complexity in the most objective manner**.

The solution

The new Process Complexity analysis comes with a multitude of KPIs that help companies measure the complexity of an arbitrary process. It heavily capitalizes on the very core of the Celonis software - the Event Log - to compute KPIs that would have been hard to compute otherwise. This is worth mentioning since the Event Log and the analyses built upon it are what makes Process Mining different from plain Data Mining and what provides Celonis with its very competitive advantage.

Whereas each of the KPIs individually looks at complexity from a different angle, they collectively provide the customer with an objective overview of the status quo of his processes. Formally, complexity here is a latent variable, and the KPIs are indicators on this variable. Therefore, we will find each of the KPIs to correlate positively with complexity.

For a given process, there is usually a multitude of paths which a single instance of the process could follow. Following our notation in subsection 1.1.1, we call different paths *variants* and different instances *traces*. Figure 2.1 below gives an example from the Accounts Receivables process. One can see that a single trace could go through a given step or not. For instance, approximately 148.500 of the investigated cases end right after the invoice is paid (see the red arrow).

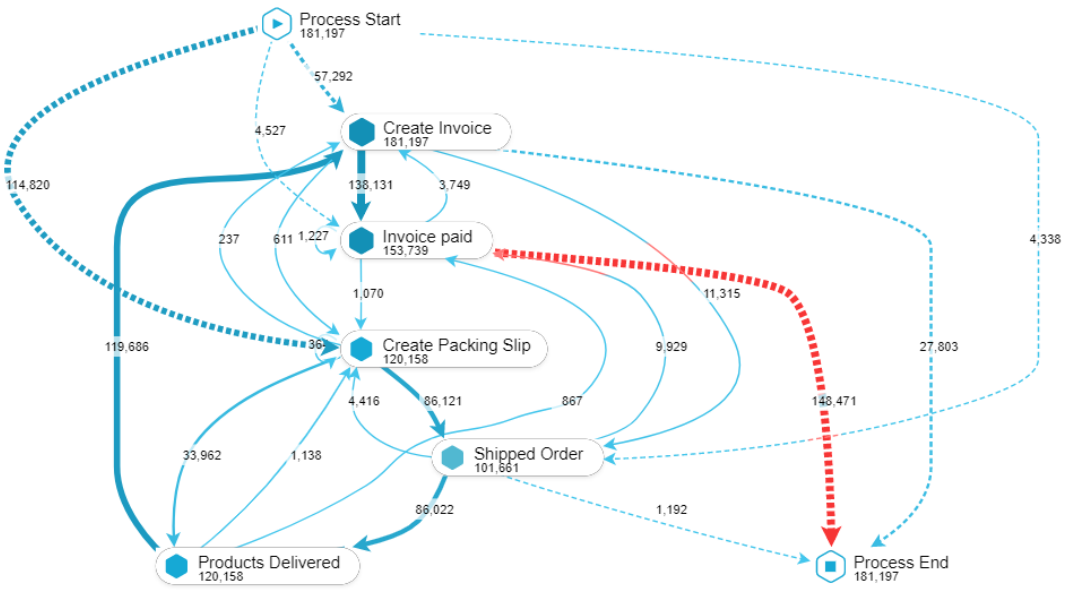


Figure 2.1: Example process with multiple variants

Looking at Figure 2.1, one can infer a first simple possibility of measuring complexity. It is in fact quite natural to claim that the more variants there are in the process, the more complex the process is (as the according graph will grow more complex). This in turn means that a more flexible company is a more complex one, which again indicates that complexity is not bad per se, since flexibility is a quality of a process rather than being a drawback. Hence, our first KPI $\tilde{v}$ is simply **the number of variants** v

available in the process, normalized by the total number of observed traces $|E|$ (see Definition 1.5), i.e. $\tilde{v} = \frac{v}{|E|}$. The normalization step is crucial for allowing for comparisons between different processes involving different numbers of traces. For example, a process with 10 traces each following a different variant (i.e. $\tilde{v}_1 = 1$) is more complex than a process with 10 variants and 2 traces per variant (i.e. $\tilde{v}_2 = 0.5 < 1 = \tilde{v}_1$).

The number of variants alone could however be misleading. It could in fact happen that all but few of these variants are rare, for example when the data at hand shows 1000 variants, 998 of which are followed only once. Although being an extreme case, this example shows that one cannot merely rely on this KPI to measure complexity. This is exactly where the second KPI, which we call **the Entropy** and denote by H, comes into play. We define the Entropy H of a process as the normalized discrete information-theoretical entropy of the frequencies of the observed variants. More precisely, the variant which a newly created trace will follow is viewed as a random variable X taking values from 1 to v , whereby X equals i if the trace follows the i-th variant. The probability $p_i = \mathbb{P}(X = i)$ for the trace to follow the i-th variant is estimated by means of the empirical frequencies $\hat{p}_i$ i.e. $p_i \approx \hat{p}_i = \frac{k_i}{|E|}$, where k_i is the number of traces following the i-th variant and $|E|$ the number of traces (cf. Definition 1.5). Thus, H is defined as:

$$H = \frac{1}{\log_2(v)} \sum_{i=1}^{v} -\hat{p}_i \log_2(\hat{p}_i) = \sum_{i=1}^{v} -\hat{p}_i \log_v(\hat{p}_i)$$

This metric shows interesting properties. In fact, if all variants are represented equally in the data, i.e. $k_i = \frac{|E|}{v}$ and thus $\hat{p}_i = \frac{1}{v}\ \forall i \in \{1, ..., v\}$, it holds:

$$H = \sum_{i=1}^{v} -\hat{p}_i \log_v(\hat{p}_i) = \sum_{i=1}^{v} -\frac{1}{v} \log_v(\frac{1}{v}) = \sum_{i=1}^{v} -\frac{1}{v}(-1) = \sum_{i=1}^{v} \frac{1}{v} = 1$$

Conversely, in the extreme case where almost all $p_i \approx 0$ but one $p_j \approx 1$ for some $j \in \{1, ..., v\}$, it holds $H \approx 0$. In fact, it can easily be proved that H ranges from 0 to 1. It is minimal whenever we have a concentrated distribution with few variants used frequently and others that are rather rare, and is maximal if every variant is equally likely. When using H as a measure of complexity, we assume that the larger H, the more complex the process is. The underlying assumption here is that a process with a large H, i.e. a process involving multiple variants with similar frequencies, is a complex one. We thus think of a non-complex process as being a process with one single variant followed almost always. This is a process that is *straightforward* to execute with few loops and decision making steps. We accordingly think of a complex process as one involving multiple steps that could be done in different orders and that depend upon multiple decisions taken by the agent(s) executing it.

A third KPI closely related to the two previous is the **average number of activities** involved in a given trace. Using the same notation as above, and defining $L(i)$ as the number of activities in the i-th variant, this KPI is given by

$$\mathbb{E}[L(X)] = \frac{1}{v} \sum_{i=1}^{v} L(i)$$

The idea behind this KPI is that a process with more activities in average is likely to be more complex as well, as it takes more steps in average to be accomplished.

The hitherto presented KPIs do not take the temporal dimension into consideration. As this could be misleading, we include two additional KPIs: **the average throughput time per trace** and **the standard deviation of the throughput time per trace**. The rationale supporting the usage of the average is quite simple: a lengthy process is likely to be complex as well, where its complexity does not necessarily stem from its structure (like the number of activities included therein), as this structural complexity need not be high, but rather from the different apriori unknown reasons causing the observed high duration. Take the example of a process that only involves two activites: receiving the invoice and paying it. In this case, the process does not look structurally complex. However, if paying the invoice takes a substantial amount of time, this could be a hint that the process is inherently complex. Similarly, the intuition behind the usage of the standard deviation is that if a process is complex, it will be likely to show a high variability in its execution time, as it is not *straightforward* to do.

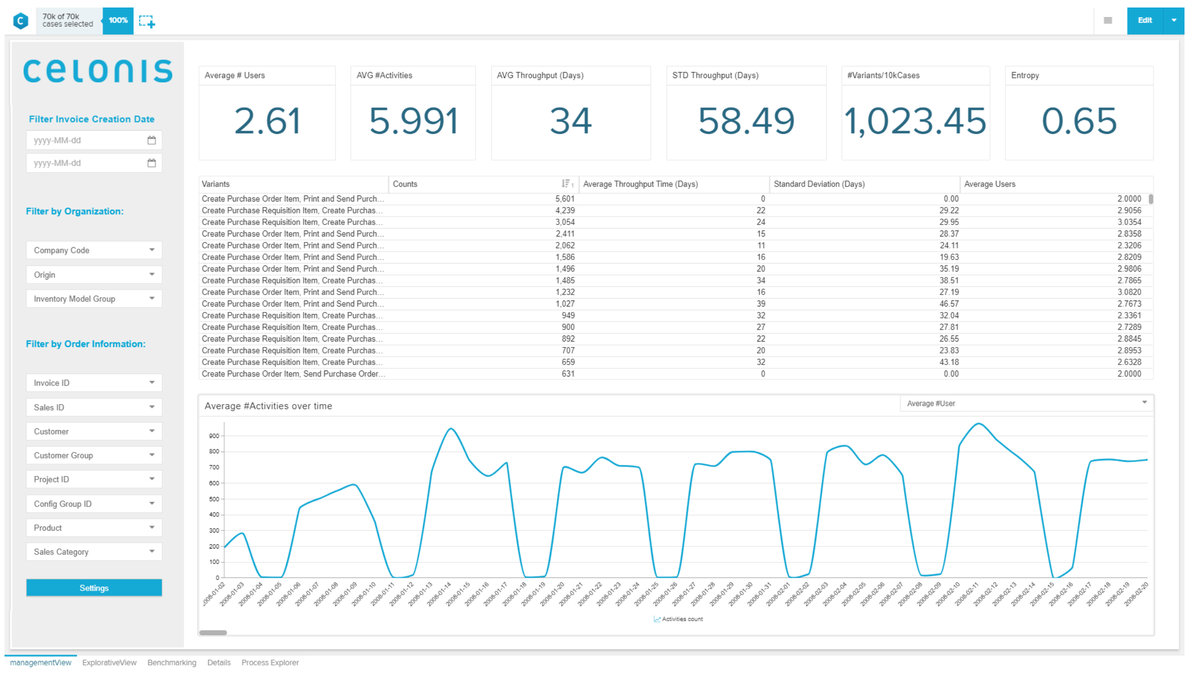
Last but not least, we use the **average number of users** as a further KPI for measuring complexity. The rationale is again simple: a process involving more human resources tends to be more complex as it apparently cannot be accomplished by a single person. This however should be interpreted carefully, since some processes involve more persons not because of their complexity but because of bureaucratic necessities (the so called segregation of duties).

As in every Celonis analysis, the Process Complexity analysis also comes with different sheets each focusing on some aspects of the analysis. For example, Figure 2.2 shows the *Management Overview* sheet, where one finds the mentioned KPIs (above), compare between the different variants with respect to their frequency, throughput time and so on (the table), and get an overview of how things developed across the time (below). The other sheets allow the customer to dig deep into the details of where things got wrong in the past. For instance, he can compare complexity when dealing with different customers, divisions of the company or products, and thus identify improvement possibilites.

2.2.3 Throughput Time Efficiency

The problem

It is totally natural that the total throughput times of some processes are higher than those of other processes, as this heavily depends on the very activities involved in each. Therefore, it is more reasonable to dig one level deeper, analyzing the throughput times between any two activities in the process. Often however, the total throughput time of a process is determined by few activities that represent its bottlenecks. While it is most effective to improve the throughput times of these very activities, chances are that these activities have already been intensively optimized, and thus already have quite efficient lead times which are hard to improve even further (for example goods delivery in a P2P process). Within this new use case, **we want to identify those very activities we**



Average # Users	AVG #Activities	AVG Throughput (Days)	STD Throughput (Days)	#Variants/10kCases	Entropy
2.61	5.991	34	58.49	1,023.45	0.65

Variants	Counts	Average Throughput Time (Days)	Standard Deviation (Days)	Average Users
Create Purchase Order Item, Print and Send Purch...	5,601	0	0.00	2.0000
Create Purchase Requisition Item, Create Purchas...	4,239	22	29.22	2.9056
Create Purchase Requisition Item, Create Purchas...	3,054	24	29.95	3.0354
Create Purchase Order Item, Print and Send Purch...	2,411	15	28.37	2.8358
Create Purchase Order Item, Print and Send Purch...	2,062	11	24.11	2.3206
Create Purchase Order Item, Print and Send Purch...	1,586	16	19.63	2.8209
Create Purchase Order Item, Print and Send Purch...	1,496	20	35.19	2.9806
Create Purchase Requisition Item, Create Purchas...	1,485	34	38.51	2.7865
Create Purchase Order Item, Print and Send Purch...	1,232	16	27.19	3.0820
Create Purchase Order Item, Print and Send Purch...	1,027	39	46.57	2.7673
Create Purchase Requisition Item, Create Purchas...	949	32	32.04	2.3361
Create Purchase Requisition Item, Create Purchas...	900	27	27.81	2.7289
Create Purchase Requisition Item, Create Purchas...	892	22	26.55	2.8845
Create Purchase Requisition Item, Create Purchas...	707	20	23.83	2.8953
Create Purchase Requisition Item, Create Purchas...	659	32	43.18	2.6328
Create Purchase Order Item, Send Purchase Order...	631	0	0.00	2.0000

Figure 2.2: Management overview from the complexity analysis

can further optimize.

The solution

The basic idea behind this analysis is not to compare the average throughput times of different activities, but to compare the different realizations of throughput time of a single activity. The assumption is that there is a high potential for efficiency increase, if the throughput time for the very same activity varies a lot each time this activity is performed: so if the throughput time of one specific activity is often far higher than the average (and often far less), we assume this activity to be inefficient, because we know that this step can be done faster. On the other hand, if the throughput time of a given activity is more or less the same every time, we assume this step to be efficient. Notice that the latter case does not mean that the throughput time is good. It just means that it could not be done faster so far. Whether this is fast or not remains a duty of the process manager.

A natural way of measuring this *within-variability* is the standard deviation:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Figure 2.3 shows the *Management Overview* sheet from the analysis, where one can see the standard deviation for the total throughput time as a single KPI, together with some other main statistics such as the minimal, average, median and maximal throughput times. On the left side is a process explorer with a new custom KPI view called STDEV. This shows the standard deviations of throughput time for all activities. On the right side, one can also see the development of the average throughput time over time, together with the corresponding standard deviation above and below the thick yellow average curve. This allows to track changes in throughput times as well as changes in throughput time efficiency. In the *Breakdown Analysis* sheet, one can additionally focus on a subprocess by selecting a start and an end activity. The left variant explorer will then show the selected cases, while the right chart will show the distribution of throughput times for the investigated subprocess. On top of the variance, this chart gives more insights about the statistics of these throughput time, e.g. whether the distribution has multiples modes, or whether it is left- or right-skewed etc... The main contribution of this analysis was thus to extend the statistics used to assess process efficiency beyond the average by including higher moments and an approximation of the underlying distribution.

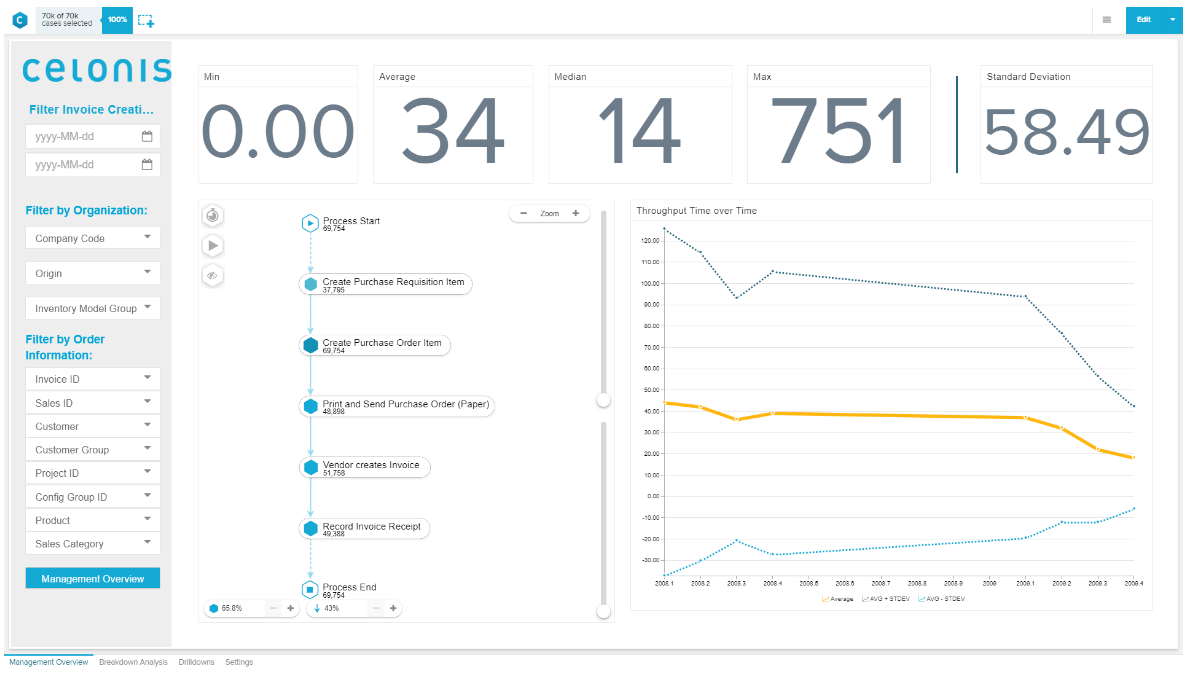


Figure 2.3: Management Overview from the Throughput Time Efficiency analysis

2.2.4 Undesired Activities

The problem

In every non- or semi-automated process, undesired activities can occur. These activities slow down the process and increase its costs. Although companies want to eliminate such activities, they often do not know where exactly to start, as they cannot precisely identify the root cause behind these undesired activities.

3 Part II: Machine Learning

The second part of this project aimed at applying Machine Learning methods on Process Mining-related use cases. Central to all these use cases was the Purchase-to-Pay (P2P) process, a standard process of nearly every existing business. Thereby, we use the definition of the standard P2P process suggested by Burns (Burns 2016), whose main activities are depicted in Figure 3.1.

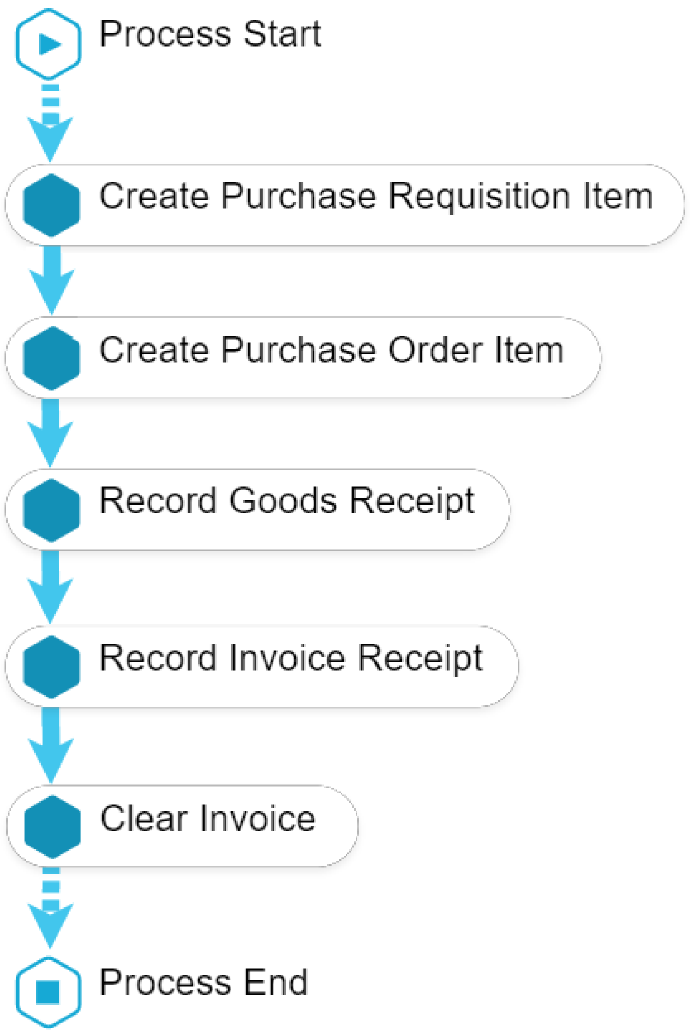


Figure 3.1: Purchase-to-Pay (P2P) process as defined by Burns 2016

For the purposes of this part, we were provided with two datasets stemming from the P2P processes. Table 3.1 provides some key facts about each of these two datasets.

Table 3.1: Purchase-to-Pay (P2P) data extracted from SAP by Celonis

	Number of events	Number of cases	Number of variants	Entropy
Data set 1	1.21 million	211,659	7,242	0.428
Data set 2	1.45 million	283,187	18,692	0.609

All utilized Machine Learning algorithms were written in Python. In order to integrate our solutions with the Celonis framework and incorporate them into stand-alone Celonis analyses, we use the Celonis Python Integration API.

3.1 Results

In this subsection we present the results of our solutions to the problem. We start by giving the hyper-parameter sets that yielded the best results for each of the presented methods individually. Subsequently, we provide a short inter-method comparison showing the different trade-offs, advantages and drawbacks of each method. Finally, we give a glimpse of the Celonis analyses we created for this part. Notice that the presented results hold for the example of the first dataset and the activity *Create Purchase Order Item*. However, similar results and conclusions were found to hold for the second dataset and/or for other activities.

4 Conclusion

With the completion of this project, several goals have been achieved.

In the *Data Analysis* part, we produced eleven Process Mining Analyses to be applied on the third most used ERP system in the world: Microsoft Dynamics AX. Second important achievement in this part was the creation of completely new analyses with a significant business decision-making value. These analyses demonstrate the capabilities of the Celonis software as they are applicable to any process and can be easily adapted to any ERP system.

In the *Machine Learning* part, we solved two Process Mining related problems. For this, we tried out different approaches and methods. The combination of our solution yielded the best results in terms of the f_1 score, achieving around 90% on the validation and testing sets.

Bibliography

Aalst, Wil van der (2014). “Decomposing Petri Nets for Process Mining –A Generic Approach–.” In: URL: `http://wwwis.win.tue.nl/~wvdaalst/publications/p721.pdf`.

— (2016). *Process mining: Data science in action.* 2nd edition. New York NY: Springer Berlin Heidelberg. ISBN: 978-3662498507.

Burns, David (2016). *Financial account in SAP: Business user guide.* First edition. Boston: Rheinwerk Publishing. ISBN: 1493213156.

Appendix

Event Log

Table 4.1: A fragment of an event log from a Purchase-to-Pay (P2P) extracted from SAP by Celonis

CASE_KEY	Properties				
	EVENTTIME	ACTIVITY_EN	USER_TYPE	NETWR	...
47141987001	14-01-2016 11:27:23	Create Purchase Order Item	A	345	...
47141987001	28-01-2016 10:34:30	Goods Receipt	A	345	...
47141987001	28-01-2016 14:51:08	Clear Invoice	A	345	...
47141987002	15-01-2016 09:11:43	Create Purchase Requisition Item	A	10000	...
47141987002	15-01-2016 09:11:52	Create Purchase Order Item	B	10000	...
47141987002	11-02-2016 10:07:16	Goods Receipt	A	10000	...
47141987002	11-02-2016 14:45:42	Clear Invoice	A	10000	...
47141987009	15-01-2016 10:01:58	Create Purchase Order Item	A	450	...
47141987009	25-02-2016 07:31:48	Goods Receipt	A	450	...
47141987009	25-02-2016 14:01:48	Clear Invoice	A	450	...
...	...	...	...	...	...