Business Information System Project

Alessandro Di Liberti

June 2024

Contents

Introduction

When it comes to deal with processes, organisation and public entities struggles to properly keep track of the flow followed by the business activities. It is is almost impossible to get a complete view of all the activities and how they interact with each other. Fortunately, the widespread use of information systems has also made it possible to collect events automatically. We can use this data to extract knowledge, measure performance, redesign, and automate processes. Techniques that allow, starting from "raw" log files, to obtain this knowledge are part of process mining.

A process mining project consists of three phases: data preprocessing, process discovery and conformance checking.

The first phase deals with cleaning up data defined as noise: data that for very often undefined reasons have no clear explanation for being included in the dataset, causing inconsistencies. Sometimes the very identification of noisy data highlights some problems within process management, or the way the information system collects the informations itself.

The second step involves the creation of a control flow model based created from the logs, this allows for a complete view of what is really going on behind the scenes. It is obviously necessary to have done proper preprocessing to avoid incorrect cases being used to build this model.

The last step is Conformance checking which by definition is a method of comparing the execution of a business process against a predefined set of conditions, such as a business process model. It is used to assess the real state of processes and to identify any variations between the should be and the as-is behavior outlined. This technique is used to determine whether the target process corresponds to the actual process, highlighting deviations between the two. In the case of process mining, it is used to assess how well the model created in the discovery phase conforms to the log file.

In this project we are going to analyze the data obtained from the information system supporting the management and processing of traffic tickets by local police forces in Italy. The system records enough data to create an event log. The register contains information on more than 150,000 traffic fines. Data were collected over a period from January 3, 2000 to June 18, 2013.

Background

The Italian police developed an information system to support the management and handling of road traffic fines by the local police force in Italy. The system records sufficient data to create an event log. The log contains information about more than 150,000 road-traffic fines.

2.1 Context

The log contains track of all activities starting with the creation of the fine, up to the time the case is closed. Thus, in addition to the activities strictly related to the administrative part, there are all the activities in case a person decides to appeal to the court. Within this analysis, the activities in the log file are partitioned into two groups: administrative part and juridical part. A more detailed description will be made later.

In general we can define a case as a sequence of activities belonging to a finite set, in this case the cardinality of this set is 11 activities. Each case has a beginning and an end activity, and almost always a sequence of activities performed within them. Each case starts with the creation of the fine and it can end either with payment, or with activities that we classify as juridical.

We aim to identify which are critical (in line with organizational goals) by observing some characteristics, such as: the sequence of activities performed, the time between activities, the duration of a case, etc.

2.2 Organizational Goals

The central office has defined its **strategic goals**, which will guide the choices made during this project.

The goals are:

1. Increasing the probability of completing the process by a payment
2. Reducing the cost of management of the process
3. Support early detection of dysfunctional executions of the process

We will now focus on each of this specific case and provide a further definition of the goals, at tactical and operational level.

2.2.1 Increasing probability of completing the process by a payment

To understand this goal, it is necessary to remember that a case can end either with payment or with an activity in the " juridical" category. An effort is made to increase the amount of cases that end with the first type mainly to reduce the lead time (time it takes to complete a case). This is because, as we will see later, cases that include "juridical" activities generally have longer processing times.

At **tactical** level we can:

- Trying to reduce the number of appeal requests
- Simplify the process of notifying and paying a traffic fine

To achieve these goals it is necessary to implement some choices at the **operational** level

- Analyze current fine management process in order to identfy any critical issues
- Test different payment methods and incentivize people to pay as soon as possible

To be precise, a measure to incentivize the payment of fines in a short period of time was taken with the "del fare" decree (*Decree Law No. 69 of June 21, 2013*) which, among other solutions to improve the Italian penalty system, introduced a 30% reduction off the total if the fine is paid within 5 days of notification.

However the data in this dataset stops on 18 June 2013, so it is not possible to see its effects.

2.2.2 Reducing the cost of management of the process

In order to reduce the cost of the process two possible paths that can be taken at the **tactical** level are:

- Automate processes
- Reduce the lead time

Both of the first two of this solutions reduce costs, the first by cutting personnel working actively on a case, and the second by reducing the time people working on a case.

For the automation solution we can, at **operational** level:

- Identify tasks that can be automated

For reducing the lead time we can apply the same operational solution we apply in order to increase the probability of completing the process by payment, since this is the fastest way to end a case.

2.2.3 Support early detection of dysfunctional executions of the process

As for this last goal, at the **tactical** level one could:

- Develop an ML model in order to identify early dysfunctional excecution
- Provide the staff alert and tools to recognize critical cases and treat them more carefully
- Simplify the path the case can take, avoiding situation that can lead to dysfunctional excecution

Both of the first two solutions require more data to be collected, and since we want also to reduce the cost, the solution including ML automation is preferable.

At **operational level** one should:

- Collect more information about the offender and the case itself in the early stages
- Identify where the path taken by the cases can be simplified

2.3 Knowledge Uplift Trail

	Input	Action taken	Output
Step 1	Raw Dataset	Structure of the data and overview of the processes	Knowledge on the data
Step 2	Raw Dataset	Filtering and identifying critical activities	-Filtered Dataset -Critical activities
Step 3	Step 2	Process discovery	Model
Step 4	Step 3	Identifying bottleneck	Area that can be improved
Step 5	Step 2	Decision tree to predict early dysfunctional excecution	-Most relevant feature -Predictor

Following here a brief explanation of this KUT:

1. The lack of an official documentation of the dataset or the process flow, made it very challenging to proceed with the analysis. So I decided to spent some times in digging into the data and trying to understand structures, flow, features...

2. The second steps consists in filtering noise, this not only produced a filtered dataset, but also highlight some activities that can cause inefficiencies.

3. In this step we obtained the model, that allow us to proceed with Step 4

4. Here we use the model previously discovered, to identifyng were possible bottleneck are located

5. In this last experimental step, I trained a decision tree classifier in order to predict the case duration based on the available feature. As a side effect we can also get which feature are more characterizing about case duration.

Dataset

The dataset is encoded in the XES standard [1].

As described in [2], the process starts with the **Create Fine** transition that writes four variables: Amount (A), Points (PO), Payment (P), and Dismissal (D).

The Amount variable refers to the amount that needs to be paid by the offender and the Points variable records the number of points that are deducted from the offender's driving license. Payment is the total amount that has been paid by the offender. Dismissal contains a character that encodes the diverse reasons for possible dismissal of the fine. A value of NIL encodes that the fine is not dismissed (i.e. has to be paid); any other value encodes different motivations.

In general, the offender can pay the fine (partly or fully) at many moments in time: right after the creation, after a road fine notification is sent by the police to the offender's place of residence, or when such a notification is received by the offender herself.

If the entire amount is paid (or, even, by mistake, more than that amount), the fine management is closed. If a notification is sent, the offender needs to also pay the postal expenses.

If the offender does not pay within 180 days, a penalty is added, usually as much as the fine's amount.

After being notified by post, the offender can appeal against the fine through a judge and/or the prefecture. If the appeal is successful, the variable Dismissal is set to value G or #, respectively, and the case ends. Otherwise, the case continues with the Receive Result activity.

If the offender does not pay, eventually the fine ends by handing over the case for Send for Credit Collection.

3.0.1 Administrative and Juridical activities

For a better organization of operations, activities were logically divided into administrative and juridical. The first category includes activities generally are carried out directly by the police or central office, for example: *Create Fine, Payment, Send Notfication....* The second partition contains activities that include operations related to the intervention of a judge or prefecture, such as: *Send For Credit Collection, Send Appeal to Prefecture, Appeal to Judge...*

3.1 Structure

The dataset is organized into rows, representing different activities, and columns representing features. Each case is identified by a group of activities (rows) that have the same value for the `case:concept:name` field. Each activity writes values for some columns, when an activity does not have to write any values, the placeholder `NIL` is used.

Methodologies

The entire analysis process was carried out on python3 environment leveraging jupyter notebooks. The project was divided into 4 macro areas, each on a notebook. Since the dataset is 561470 rows × 16 columns, to speed up the process of loading data into each notebook I used serialization with pickle to transform the dataset into a .pkl file

The macro areas are:

- **Data Overview**
- **Variant Analysis and Filtering**
- **Process Discovery and Conformance Checking**
- **Machine Learning Prediction**

4.0.1 Software used

The following software was used:

- Python3
- generic data manipulation libraries, such as `Pandas` [3], `numpy` [4]...
- Lib `pm4py` [5], to compute process mining operations
- PMTK [6], a software used to have a graphical representation of information about the dataset
- Lib `scikitlearn` [7], in order to create the decision tree classifier

4.1 Data Overview

As mentioned earlier, the lack of official documentation explaining the structure of the dataset, the meaning of the features, and the behavior of the processes; made it complicated to continue with more in-depth analysis. For this reason, I chose to use PMTK in combination with pm4py, to obtain some information such as:

- Start Start and end activities
- Missing values
- Which attributes where written by which activities
- Frequencies for each activities

4.2 Variant Analysis and Filtering

This macro area consists of two vital activities for the purposes of a process mining project: variant analysis and filtering. The way the analysis was conducted it was decided to keep both of these operations in the same segment, in fact we used the knowledge generated by the variant analysis to filter noise and from the filtered data we conducted a new analysis on the traces. At the end of this process we obtained some useful goal information and a filtered dataset to be used in the discovery phase to generate a process model.

4.2.1 End activities

The first operation was to filter out all the cases that end with a non compliant activity. There are in fact only four way a case can end:

1. Payment
2. Send For credit collection
3. Send appeal to prefecture
4. Appeal to judge

4.2.2 Case duration

The second step was to filter the case with duration equals to zero, and then perform a graphical representation of the distribution duration of the cases.

After applying some constraint on the variant, I proceed to analyze the first 25% of case by duration, and the cases from 75%. This was done in order to understand which feature were related to, respectively, short and long cases.

4.3 Process Discovery and Conformance Checking

The goal of this section was to create a representative model of processes by reconstructing it from the log file. To do this, four process discovery algorithms, offered by the pm4py library, were used, and then conformance checking operations were computed to assess which model was most consistent with the log file.

The algorithms generate a Petri net, a mathematical modeling tool used to describe and analyze the flow of information and control in systems. It consists of places, transitions, and arcs connecting them

We use conformance checking to assest the quality of the generateed petri net. In particular we use token based replay and alignment techniques.

Token-based replay consists in comparing an event log with a process model by simulating the execution of the model using tokens. Tokens represent the state of the process, moving through the model as events occur, to verify if the log aligns with the expected behavior.

The alignment technique for conformance checking compares an event log with a process model by finding the optimal alignment between logged events and model activities. It identifies discrepancies and deviations by aligning each event to the corresponding model element, ensuring minimal cost of mismatches.

To decide whether a miner is good we look at the metrics produced by the conformance checking:

- **Fitness** [8]: the discovered model should allow for the behavior seen in the event log
- **Simplicity** [9]: the discovered model should not be unnecessarily complex
- **Precision** [10]: the discovered model should not allow for behavior completely unrelated to what was seen in the event log
- **Generalisation** [11]: the discovered model should generalize the example behavior seen in the event log

In the best case we would take the model with the higher values for each metric. Most likely this won't happen, and so we will focus on the one with an overall high score, and a high level of precision, and preferably high simplicity. This is because we want to identify any bottleneck or inefficient situation and we want a model that doesn't allow any behavior that was not seen in the log.

4.3.1 Algorithms and models

Alpha miner

The first algorithm used was the alpha miner [12]. The Alpha Miner algorithm constructs a Petri net from an event log. It identifies causal relationships by examining the direct succession of events and determines the parallel, sequential, and concurrent relationships between activities. The algorithm infers places to represent conditions and connects these places with transitions based on observed sequences. This results in a model that captures the structure and behavior of the process, including parallelism and loops.

ILP miner

The ILP Miner [13] is a process discovery algorithm that uses Integer Linear Programming (ILP) to construct a Petri net from an event log. Integer Linear Programming is a type of optimization problem where the variables are integer values and the objective function and equations are linear.

The algorithm formulates the discovery task as an optimization problem, aiming to find a model that maximizes the fitting of the log while adhering to process constraints. The ILP Miner can handle complex dependencies and produce more precise models, but it requires significant computational resources and may not scale well with large event logs.

It generates perfect fitness, with high generalization, but precsion and simplicty are low.

Inductive miner

The Inductive Miner [14] is a process discovery algorithm that constructs process models using a divide-and-conquer approach. It divides the log into smaller sublogs based on directly-follows relations, identifies the most significant process patterns (e.g., sequences, parallelism, choices, loops), and recursively applies this procedure to each sublog. This method ensures the generation of sound process models, capable of handling noise and producing comprehensible and structured models.

Heuristic miner

The Heuristic Miner [15] is a process discovery algorithm that creates process models from event logs by leveraging frequency-based heuristics. It identifies the most frequent directly-follows relationships between activities and uses these to construct a dependency graph. The algorithm applies thresholds to filter out less significant relationships, reducing the impact of noise and infrequent behavior. This results in a more robust and simpler process model, suitable for handling real-world logs with irregularities.

4.4 Machine Learning Prediction

The last part of this project involves the use of a decision tree classifier [16] in order to predict the duration of a case given some initial information. This type of classifier was chosen for three reasons: its good prediction ability, the simplicity of reading the generated structure and the possibility of identifying which features contributed most to the classification.

The definition of the problem is the following: given ten attributes for each case, predict the duration of the latter, divided into four possible classes.

- **short**: < 7 days
- **medium-short**: < 3 months

- **medium-long**: <= 1 year
- **long**: > 1 year

Two versions were produced, changing the structure of the training set.

4.4.1 ML v1

In this first version, ten attributes were used that could be obtained in the first three days after the creation of the dataset. these attributes are:

- amount: the amount to be paid
- article: the article corresponding to the transgression committed
- ac1 - act2 - act3 - act4: these are four features representing the action taken in the first three days. If in the three days since creation, only the first activity occured, or less than four, a 'missing' value is placed for the remaining activities.
- org:resource: the resource handling the creation of the case
- day - month: the day and the month the fine was created
- vehicleClass: the class of the vehicle

The idea behind using 'missing' value for the missing activities is to try to leverage the fact that few action were taken in the first three days, and this may can help to prediction.

The day and month were chosen beacause it appear to be some sort of correlation between an increase of the number of fines and particular time of the year. As shown in fig 4.1, the Create Fine activity has some spike in what it's seems to be the third quarter of each year.

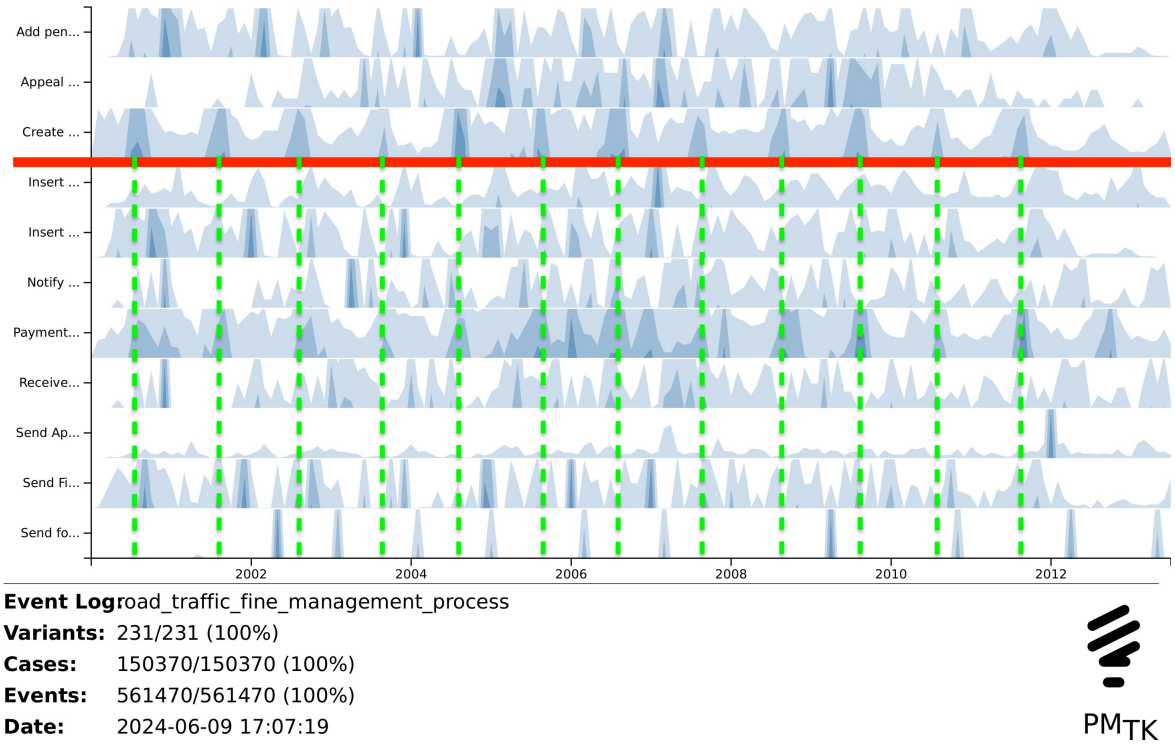

Figure 4.1: Horizon chart of the activities produced with PMTK

Unfortunately this solution achieved bad result, thus I proceed with the second version.

4.4.2 ML v2

In this version, the main goal was no longer to predict duration, although the classifier was trained for this, but to identify which features were most characteristic of a case's duration. The four activities in the first three days were removed and columns were added:

- act2: the second activity occuring in the case, only if it happens within three months; this was done to keep the dataset balanced.
- deltaTime: the time difference between first and second activity.

Seen the good result on a first attempt, I improved the training with the following solution:

1. **One-hot encoding**: a technique used to convert categorical data into a binary matrix representation. Each category is transformed into a vector where only one element is "1" (hot) and all others are "0".
2. **Hyperparameter tuning**: A decision tree classifier can achieve higher score if its hyperparameter (*splitting criterion, max depth,...*) are tuned. The grid-search [17] technique systematically explores a specified set of hyperparameters. It evaluates the model's performance for each combination of hyperparameters in the grid, identifying the best configuration based on a chosen evaluation metric. In a more complex model this approach is computationally very expensive, but in this scenario the best classifier was found in just a few minutes.
3. **5-fold cross validation** [18]: this technique consists in dividing the dataset in k (5 in this case) folds, the model is trained with $k-1$ folds and the remaining fold is used for the testing. The process is repeated k times, each time using a different fold for the testing. The measure are obtained by averaging.

Results

In this chapter, we apply the methodologies seen in the previous chapter and show the results for the following points

- General knowledge of the cases
- Variant and filtering analysis
- Process discovery
- ML prediction

5.1 General knowledge of the cases

In this section we will show some results and statistics.

In fig 5.1 we can notice how 95% of variants are structured. We see that every case starts with "*Create Fine*" and can end in different ways; but as we mentioned earlier there are only four activities a case can end with.

In this plot it's interesting to notice how a good portion of cases end with "*Send Fine*".

This activity has repeatedly emerged as one of the most problematic. This can be seen well by analyzing the plots of case duration, which we will see later.

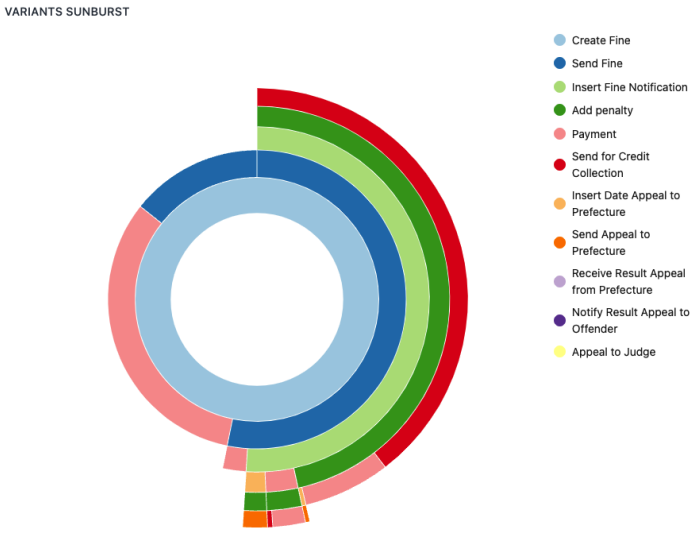

Figure 5.1: Top 95% of most frequent variants

Another interesting chart to look at is fig 5.2; we can see that the goal of increasing the probability of a case ending in "*Payment*" is well-founded. In fact, currently only 51% of cases end in this way; while 39% end with "*Send for credit collection*" which greatly increases the lead time.

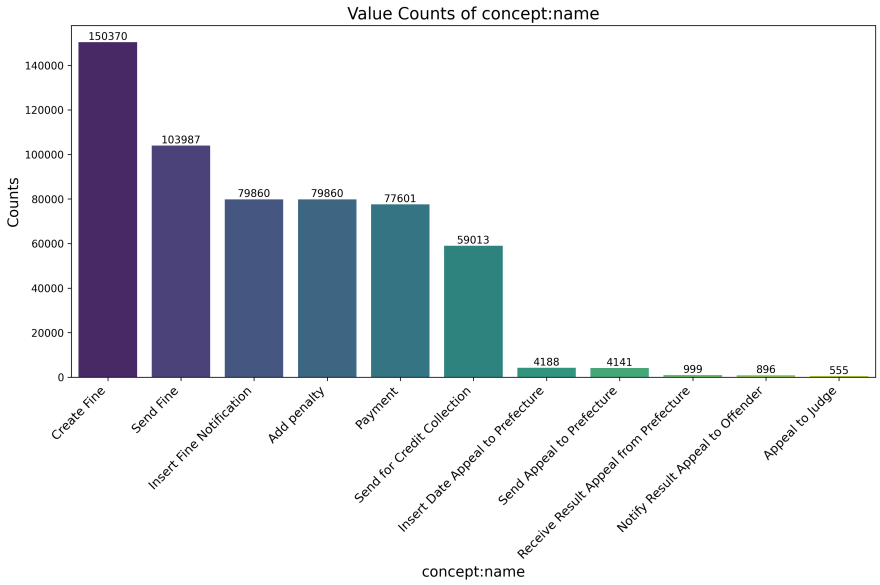

Figure 5.2: Frequency for each activities

The last chart is not strictly relevant to the goals set, but it shows information that for future use can be investigated further. In particular fig 5.3, shows a plot of which offenses are most commonly committed (indicated by the article) and it can be seen that three are clearly more prevalent. Articles 157, 7 and 158 alone compose the majority of infractions and refer to situations where the vehicle is parked in a prohibited place

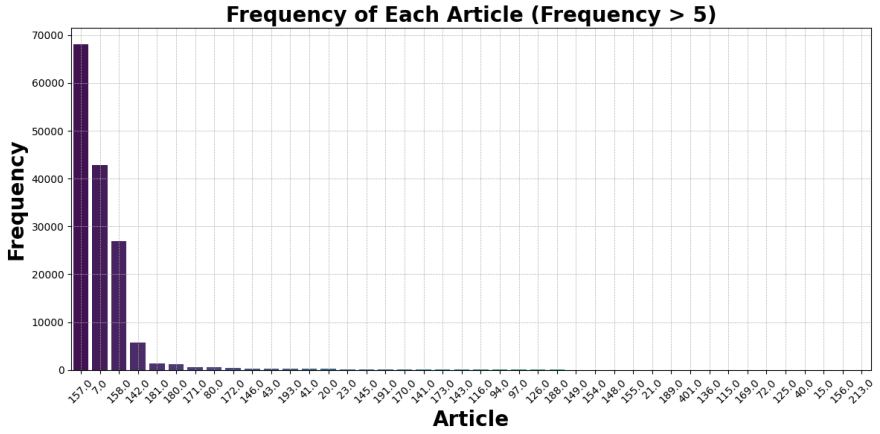

Figure 5.3: Frequency for each article in the dataset

5.2 Variant and filtering analysis

In this section we perform filtering and variant analysis. The first step was a filtering of end activities. In general, strong intakes were avoided throughout the filtering process. The motivations behind this choice are mainly twofold: we do not have precise domain knowledge, and making assumptions about what is noise and what is not might remove some important observations; the second motivation is that we try to maintain any anomalous behavior so that it is visible in the graphical model

Before actually removing cases that do not complaint with the constraint of ending with one of the four allowed cases, we conducted an analysis on the instances ending with "*Send fine*", which are 20755. One possibile explanation of this high number is the fact the fine was sent beyond the 90 days

established by law, therefore invalidating it. But after computing this check we can see that only 11'942 cases have an explanation for their behavior. For the remaining cases, since we don't have any documentation about it, we will consider them as anomalies and remove them. I decided not to remove the cases ending with "*Send Fine*" that has < 90 days duration, because even if it's not a wanted behavior, this does not qualify as noise; but it is something to keep in mind when we analyze the Petri net model.

Another filtering was done for cases that end with an appeal and have the value NIL in the dismissal field. NIL means that the appeal was not won, and therefore it is not possible for a case to be terminated in this way.

The final filtering was done on all the other ending activities.

At this point we filter out about 10 thousand cases, remaining with 140176 cases.

The second step of this filtering was based on case duration. In particular we first analyze the Cumulative Distribution Function of case duration 5.4.

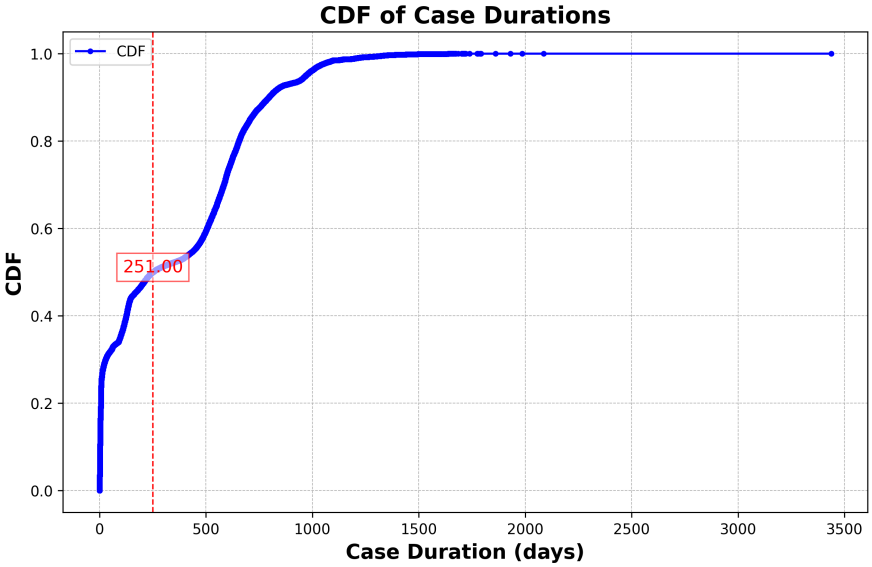

Figure 5.4: CDF of case duration

This plot (5.4) highlights the fact that there were cases with duration equals to zero, in particular, 4802 cases. Variants of this group are shown in fig 5.5

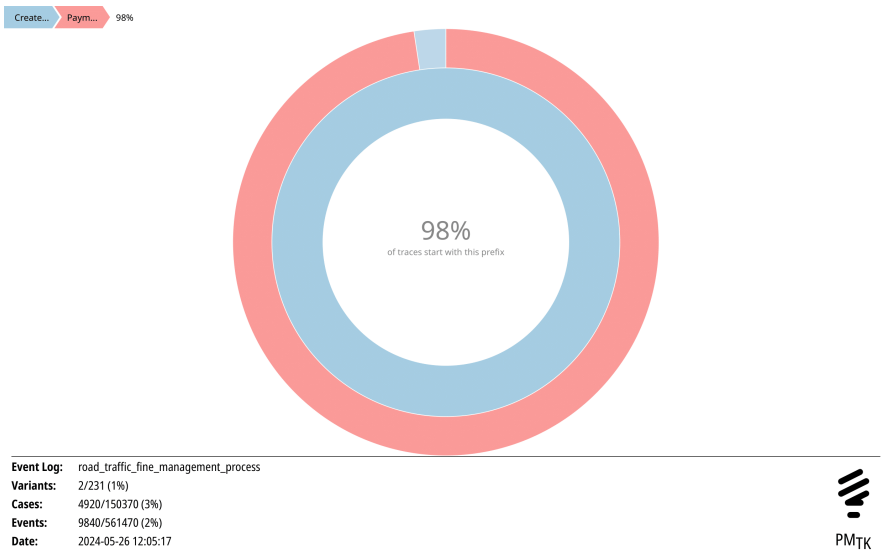

Figure 5.5: Sunburst plot of variants of cases with duration 0

As we can see 98% of cases follow the same trace, "*Create Fine*" $\rightarrow$ "*Payment*", we can consider this cases as "manually" resolved by an operator. But since we cannot extract any information about the process we can discard this cases, as well for the 2% remaining that we just mark as noise.

In the end I applied constraint on the order of activities, obtained from a little domain knowledge, and filtered out 1030 cases.

The total number of case left in the filtered dataset is 134346.

5.2.1 Statistics on filtered log

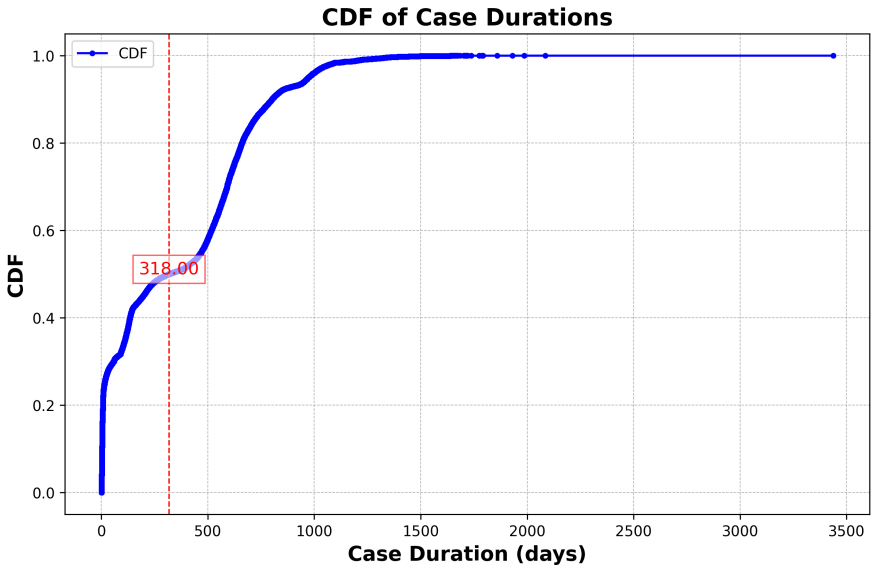

Figure 5.6: CDF of case duration after the filtering

After the filtering we have a new CDF of the duration of case duration, fig 5.6.

Although more precise analyses are needed, it is still possible to assume that the distrubution ideally follows a powerlaw or lognormal: i.e. long-tailed distributions in which most cases are described by a short duration, but exceptions are still present.

The quartiles are:

- first quartile: 15 days
- second quartile: 318 days
- third quartile: 623 days

We now focus on the analysis of the first and last quartiles. Looking at how the distributions change and analysing the variants present.

Short cases

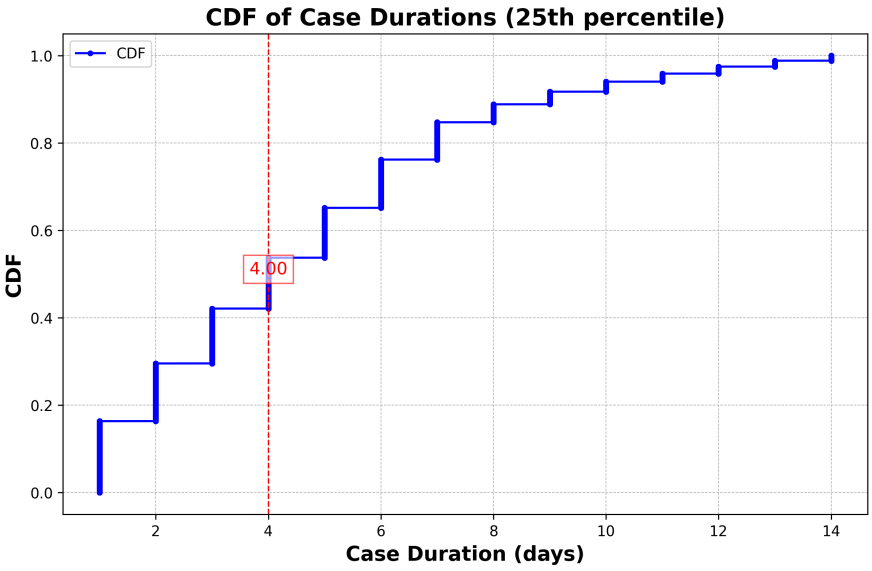

Figure 5.7: CDF of short case duration

Looking at this distribution, fig 5.7, we can see that half of the cases resolve within four days. We must keep in mind that cases with zero duration have been eliminated, so we are looking at realistic behavior. Analysing the variants we have that 99% follow: *Create Fine* $\rightarrow$ *Payment*.

This indicates that a case to be short must have as few activities as possible. Even the Send Notification activity is not present. We can therefore identify a short case as one in which the person pays directly without waiting for notifications. Wanting then to decrease the lead time, the government's incentive to pay in the next 3 days could lead to results.

5.2.2 Long cases

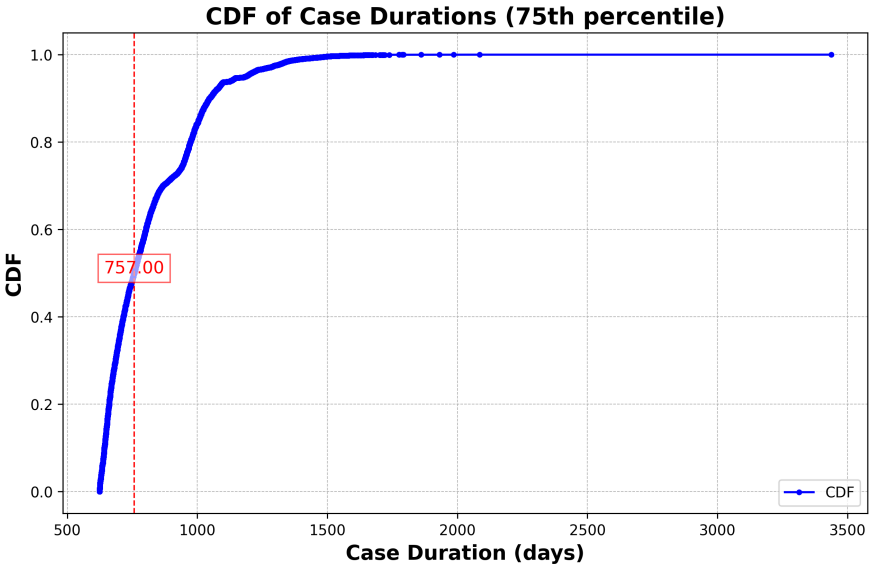

Figure 5.8: CDF of long cases duration

Looking at the distribution of long cases, fig 5.8, we note how this tends to follow a heavy-tailed distribution, suggesting that most of the cases are described by the same variant. Indeed, looking at the frequency of variants present, we have that:

- 92% of cases follows: *Create Fine* → *Send Fine* → *Insert Fine Notification* → *Add Penalty* → *Send For Credit Collection*
- 4% of cases follows: *Create Fine* → *Send Fine* → *Insert Fine Notification* → *Add Penalty* → *Payment*
- The remaining are just noise for the purpose of this analysis

This finding is very interesting as it highlights two points: when the credit collection is involved the duration of cases is much higher, and the juridical part (making an appeal) does not slow down the process as much the credit collection does.

There is a possible explanation behind the duration being extend with the credit collection. We use the PMTK dotted chart to have a view of what is happening.

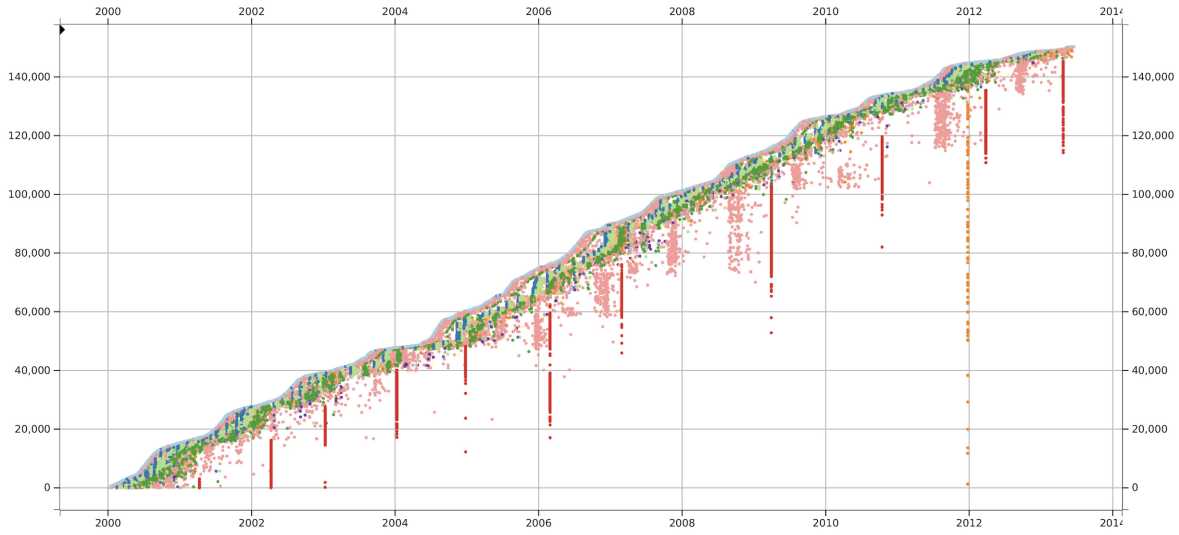

Figure 5.9: Dotted chart of actvities

In fig 5.9, The red dots represent the activity *Send for credit collection*. We can see how this has an annual cadence, generally the end of the year. So each case (that needs credit collection) must necessarily wait until the day of the year when the credit fine collection is set, in order to close the procedure.

Another two comments we can make on this chart are:

- The pink dots represent the activity *Payment*, here we can see that the payments are more frequent some days before the date set for the credit collection. This is just an assumption and this event can be caused by some other factors, unfortunately, not having a proper documentation blocks us to explain every behavior.
- On 2012 something happen with the *Send appeal to prefecture*, represented by the orange dots, all of this activites are set on 25th of december 2012. Again, not having a proper documentation don't help us in explain this event.

5.3 Process discovery

In the process discovery part, four algorithms were used to create a Petri net-based model. Metrics of Precision, Fitness, Simplicity and Generalization were used to evaluate which model was most consistent with reality. The main quantity we are looking to maximize is Precision, but also having high Simplicty is preferable.

5.3.1 Alpha Miner

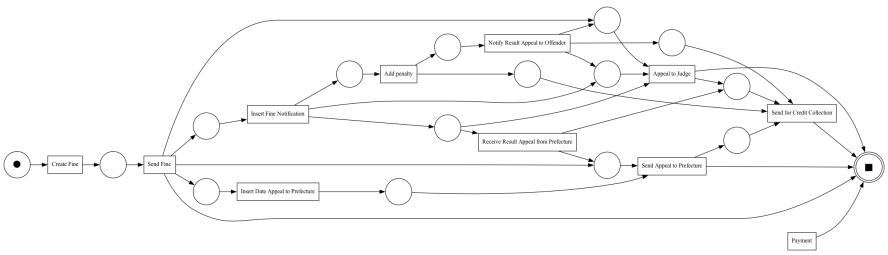

Figure 5.10: Petri net by Alpha Miner

Score of Alpha miner model, fig 5.10:

- **Precision**: 0.7642
- **Fitness**: 0.6813
- **Simplicity**: 0.5294
- **Generalization**: 0.9631

This algorithm produced overall low scores.

5.3.2 ILP Miner

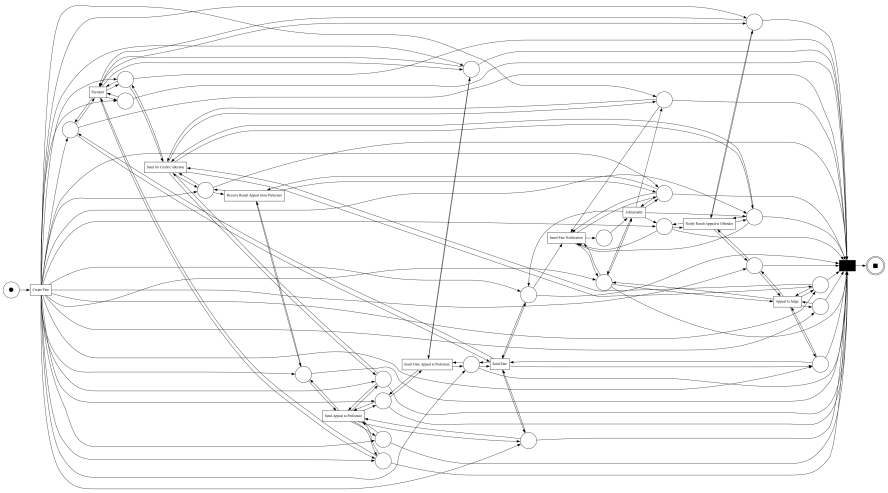

Figure 5.11: Petri net by ILP Miner

Score of ILP miner model, fig 5.11:

- **Precision**: 0.5572
- **Fitness**: 1.0
- **Simplicity**: 0.1544
- **Generalization**: 0.9660

Precision and Simplicity are too low.

5.3.3 Inductive Miner

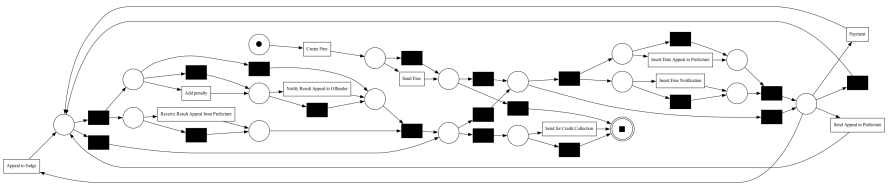

Figure 5.12: Petri net by Inductive Miner

Score of Inductive miner model, fig 5.12:

- **Precision**: 0.5048

- **Fitness**: 1.0
- **Simplicity**: 0.6000
- **Generalization**: 0.9640

This model has perfect fitness, and this lead to having low Precision.

5.3.4 Heuristic Miner

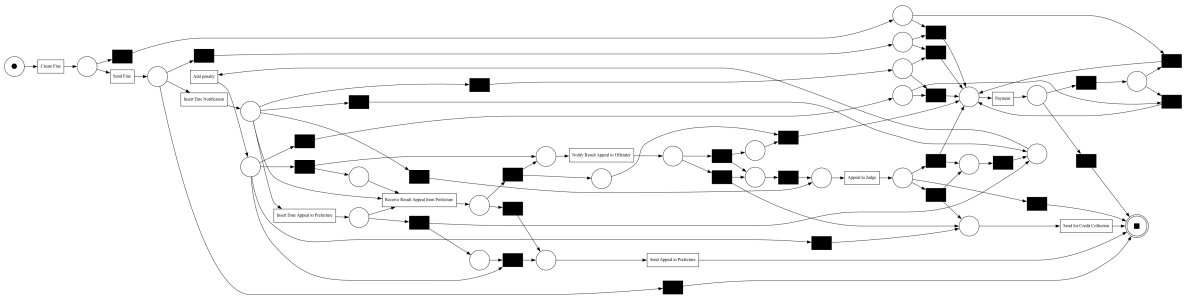

Figure 5.13: Petri net by Heuristic Miner

Score of Heuristic miner model, fig 5.13:

- **Precision**: 0.9962
- **Fitness**: 0.9158
- **Simplicity**: 0.5537
- **Generalization**: 0.8489

This is algorithm produced the model with the scores we want. So we will use this to conduct some analysis.

Alignment

Alignment-based replay aims to find one of the best alignment between the trace and the model. For each trace, the output of an alignment is a list of couples where the first element is an event (of the trace) or $\gg$ and the second element is a transition (of the model) or $\gg$. For each couple, the following classification could be provided:

- **Sync move**: the classification of the event corresponds to the transition label; in this case, both the trace and the model advance in the same way during the replay.
- **Move on log**: for couples where the second element is $\gg$, it corresponds to a replay move in the trace that is not mimicked in the model. This kind of move is unfit and signal a deviation between the trace and the model.
- **Move on model**: for couples where the first element is $\gg$, it corresponds to a replay move in the model that is not mimicked in the trace. For moves on model, we can have the following distinction:
  - **Moves on model involving hidden transitions**: in this case, even if it is not a sync move, the move is fit.
  - **Moves on model not involving hidden transitions**: in this case, the move is unfit and signals a deviation between the trace and the model.

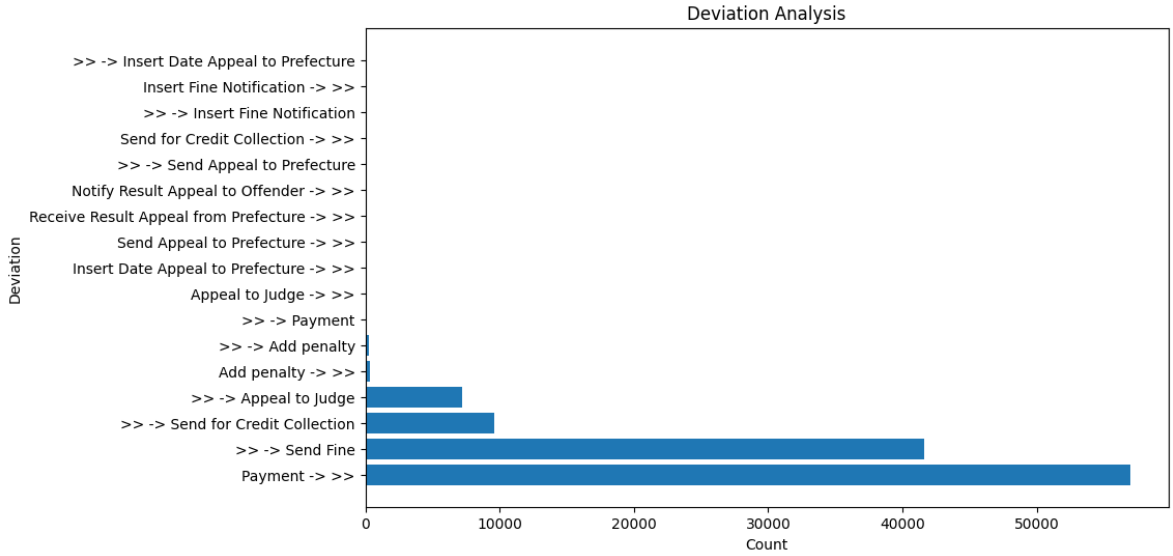

Figure 5.14: Deviation analysis on Heuristic Miner generated model

After running alignment conformance checking we produced the list of aligned traces, this produced the following results:

- **Total Deviations**: 116578
- **Log moves** (event in log not in model): 58828
- **Model moves** (events in model not in log): 57750

Looking at the frequencies of the most common deviations, fig 5.14, we can explain this behavior and extract some useful knowledge:

- ***Payment*** $\rightarrow$ >> : This is the easiest to explain because the model permit another action after the payment, while in reality this occurs rarely. There are some case where the fine was not fully paid and so the person continued to pay until he reached the total amount to be paid.
- >> $\rightarrow$ ***Send Fine***: This deviations is a problematic already appeared in the preliminary analysis. We know that the police has a problem with the notification sending, causing the creation of a lot of variants. When it comes to Send Fine we know that the only accettable previous activity is Create Fine, but instead we have a lot of different activities that the model cannot capture.
- >> $\rightarrow$ ***Send For Credit Collection*** and >> $\rightarrow$ ***Appeal to Judge***: This two deviations, being related to two accettable end activities, can be reached from multiple different activities, thus creating a lot of deviations.

5.4 ML prediction

The goal of this section is to test whether a classifier can predict the duration of a case with the available data. A side bonus is that since it is a tree-based classifier we can gain insights into which features are most relevant to the decisions made

The first version of the predictor did not perform well. This version, in which only data available at the time a fine is created is used, does not allow the duration of a case to be predicted. In particular we obtained:

- **Accuracy**: 61.93%
- **Precision**: 65.74%

- **Recall**: 61.93%
- **F1-score**: 52.76%

In the second version, however, the aim was more to identify more relevant features. By varying the dataset and using the first two tasks with the delta time between them, we were able to obtain higher scores. Encoding, hyper parameter optimisation and cross validation techniques were also applied in this version. The scores obtained are:

- **Accuracy**: 89.49%
- **Precision**: 84.03%
- **Recall**: 89.49%
- **F1-score**: 86.20%

Obviously these values are not sufficient to be able to use the classifier, but they give us enough confidence to observe which features are most relevant.

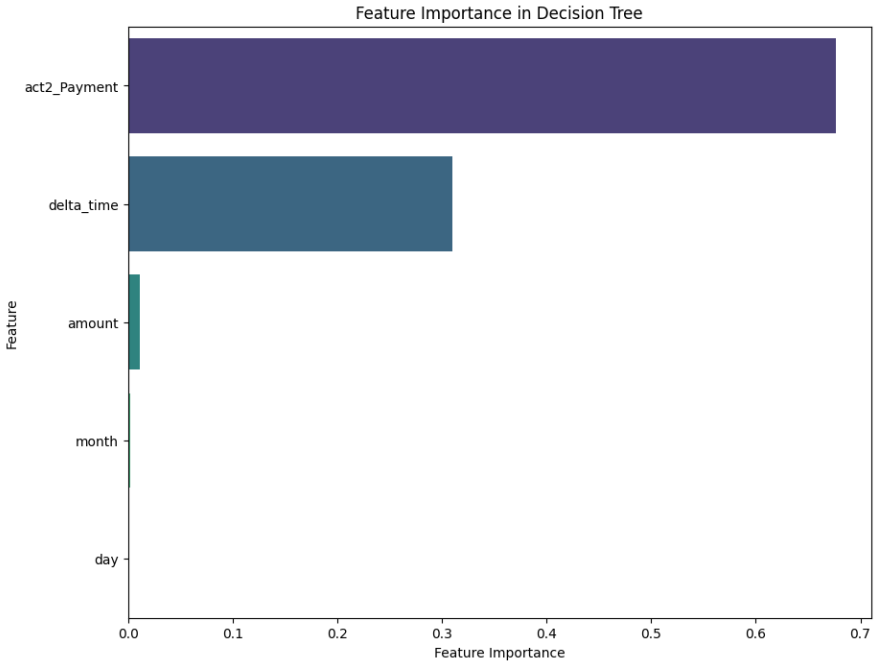

Figure 5.15: Importance of feature of classifier trained with ML v2

In fig 5.15, the *Payment* activity is obviously the most important one, since about half of the cases in the training dataset have just two activities and the second is *Payment*, which is also an end activity.

More interesting is *delta-time*, that may suggest that the waiting time between the first and the second activity can be an hint of the lead time of the case. However, this assumption would require more analysis since the importance of this feature could be linked to cases that have payment as the terminal activity.

Much more interesting is instead the feature amount. This suggest that the amount of the fine to be paid is sometime an hint of the lead time. This result may suggest that the incentive program launched by the government could be successful.

Conclusion

In this project, classic process mining strategies were applied to achieve the goals set by the central office. The results of this analysis indicate several solutions that can be used to improve the processes of fine management.

Relative to the goal of **increasing the probability that a case is completed with a payment** we observed that: most of the cases, which can be considered short, end with payment.

Thus, the proposed solution is to provide an incentive to process cases as quickly as possible (a maneuver that we recall was implemented in June 2013).

The second goal is **Reducing the cost of management of the process**, to achieve it has been proposed to try to automate processes that would decrease lead time. Since most juridical operations are difficult to automate, the only real proposal is to automate, or completely overhaul, the fine notification. We have observed multiple times how *Send Fine* is the cause of many problems that slow down the process and generate unintended variances. We observe how very often this is not sent within the 90 days, thus causing the fine to be cancelled. It is also happens that an offender does not realize that he or she has received a fine, and the delay in notification slows down the process by a lot; in addition, it must be considered that with the sending of the notification, fees are charged, and can disincentivize a person from paying a fine.

Therefore, the proposal is to automate the sending of the notification, at the very moment the fine is created, taking advantage of telematic channels, so as to avoid charges on the citizen. In case of non-payment within n days then a classic postal notification is send.

The last goal is **Support early detection of dysfunctional executions of the process**, and we proposed to implement machine learning model to help this detection, or to simplify the path of excecution. The latter requires a domain expert to help the redesign the process, while the ml solution requires more feature to be collected. We have in fact observed how the lack of features does not allow for efficient training. The first proposed version showed how the data available at the time of fine creation were not sufficient to identify the duration of a case. The second version shows how most of the features present do not contribute to the final decision.

Therefore, the solution that is proposed is to increase the number of features available at the time of creation, these can include information about the offender that would obviously need anonymization; but also information during different activities. Combining this and a staff labeling of dysfunctional cases, it would then be possible to train a model that achieves the proposed goal.

Bibliography

[1] C.W. Gunther and H.M.W. Verbeek. *XES - standard definition.* BPM reports. BPMcenter. org, 2014.

[2] Felix Mannhardt, Massimiliano De Leoni, Hajo A Reijers, and Wil MP Van Der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98:407–437, 2016.

[3] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[4] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

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

[6] Gabriel Steinhardt. *Definition of Product Management.* 05 2017.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Alessandro Berti and Wil MP van der Aalst. Reviving token-based replay: Increasing speed while improving diagnostics. *ATAED@ Petri Nets/ACSD*, 2371:87–103, 2019.

[9] Borja Vázquez-Barreiros, Manuel Mucientes, and Manuel Lama. Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm. *Information Sciences*, 294:315–333, 2015.

[10] Jorge Munoz-Gama and Josep Carmona. A fresh look at precision in process conformance. In *International Conference on Business Process Management*, pages 211–226. Springer, 2010.

[11] Joos CAM Buijs, Boudewijn F van Dongen, and Wil MP van der Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, 23(01):1440001, 2014.

[12] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE transactions on knowledge and data engineering*, 16(9):1128–1142, 2004.

[13] BF van Dongen and WPM Nuijten. Process mining using integer linear programming.

[14] Sander JJ Leemans, Dirk Fahland, and Wil MP Van Der Aalst. Discovering block-structured process models from event logs-a constructive approach. In *Application and Theory of Petri Nets and Concurrency: 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings 34*, pages 311–329. Springer, 2013.

[15] Anton JMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristicsminer algorithm. 2006.

[16] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[17] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[18] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36(2):111–133, 1974.