Road Traffic Fine Management Process

Bianchessi Mattia

July 2024

Contents

1 Introduction

This project (paper) aims to present an analysis of business processes using Process Mining techniques. This project aims to exploit process mining techniques to find some optimizations for the management of processes involving traffic tickets. The first chapters present the log and describe some cleaning and filtering operations to keep the data meaningful for the purpose of analysis. Then the log is analyzed through statistical methods to find areas for improvement, analyzed variants. Finally, business objectives to improve business efficiency are presented.

2 Method

The method here used is The Knowledge Uplift Trail (KUT) that turn raw data into useful insights. The method consist of in a series of steps that eventually provide a clearer understanding of the data in the dataset. The technique used are:

1. **Data cleaning**: prepare the event log make it more human readeable, i.e. filling missing value, standardizing some column, and some convertions.
2. **Data filtering**: cut out some invalid cases, filter out the noise to keep the relevant information.
3. **Descriptive analysis**: here it uses some statistical features to analyse some characteristics.
4. **Process mining**: use process mining techniques to discover the process.
5. **Conformance checking**: identifies any deviations from the typical sequence of event.
6. **Intervention strategies**: are designed based on the insights from the previous steps and they are continuously improved through monitoring and measurement.

2.1 Dataset

This section provides an overview of the dataset that are used for the studies. First of all we have a look through the different attributes of the raw dataset[1]. It also shows a head of the table.

- **amount** contains the monetary amout value that needs to be paid by the offeder.

[1] missing description cause by the unexisted specifications

- **dismissal**indicates whether the fine was dismissed. NIL means the fine was not dismissed, while # or G indicate that the fine was dismissed by a judge or by the prefecture.
- **concept:name**: describes the specific activity related to the process(i.e "Create Fine," "Send Fine," or "Payment").
- **vehicleClass**: specifies the class of the vehicle involved in the fine.
- **time:timestamp**: records the exact date and time when each event occurred.
- **article**: refers to the regulation under which the fine was issued.
- **points**: shows the number of points deducted from the driver's license.
- **case:concept:name**: identifies the unique case to which the events belong.
- **expense**: an additional expenses associated with the "Send Fine" event.
- **paymentAmount** records the amount paid by the offender in a "Payment" event.

Missing description

- **org:resource**: identifies the organizative resourse that take care of the case.
- **totalPaymentAmount** records that show the total amout paid by the offender till this moment.
- **lifecycle:transition**: This attribute represents the transition in the life cycle of an activity[2].
- **notificationType**: This attribute indicates the type of notification sent to the offender.
- **lastSent**: This attribute records the date and time of the last notification or communication sent to the offender regarding the fine.
- **matricola**: This attribute represents a unique identifier assigned to a police officer or operator within the traffic ticket management system[3].

[2]Dataset contains only complete for this column
[3]Every case has a '0' value of matricola attribute or Nan

```
def fixAmount(row):
    if row['concept:name'] == 'Create Fine':
        return row['amount']
    elif row['concept:name'] == 'Send Fine':
        return row['expense']
    elif row['concept:name'] == 'Add penalty':
        return row['amount']
    elif row['concept:name'] == 'Payment':
        return row['paymentAmount']
    else:
        return 0

def replace_nan_with_zero(log):

    columns_to_check = ['amount', 'expense', 'concept:name', '
        totalPaymentAmount', 'paymentAmount']

    for column in columns_to_check:
        if column in log.columns:
            log[column] = log[column].fillna(0)

    return log


log_raw['amount'] = log_raw.apply(fixAmount , axis=1)
log_raw = replace_nan_with_zero(log_raw)
```

Code 1: Fix amount column

3 Data cleaning

This section present some data manipulation to make the dataset more human readabble.

1. It is necessary to consolidate the values of the *amount*, *expense*, and *paymentAmount* columns into a single amount column in order to simplify the analysis of transaction data in the event logs and make the log more readable. In addition, the *Nan* values of these columns have been replaced with 0s.

2. To increase clarity some values in the *dismissal* column are modified by removing the *Nan*, *Nil* and # values while maintaining a textual representation of the case.

3. Cases where it is unclear whether the case is not finished are filtered out to analyze concrete cases.

3.1 Consolidate amount column

- The *fixAmount* function determines the correct value of the amount associated with a particular activity within the process. It returns the specific

```
def readableLog(row):
    if row['dismissal'] == '#':
        return 'Pr' # 'Send Appeal to Prefecture'
    elif row['dismissal'] == 'G':
        return 'Jd' # Appeal to Judge
    elif row['dismissal'] == 'NIL':
        return 'NP' # Non ancora pagata
    elif pd.isna(row['dismissal']):
         return 'MV' #Missing value
    else:
        return row['dismissal']

def set_terminate(row):
    if row['dismissal'] in ['#', 'G']:
        return 'Yes'
    elif row['dismissal'] == 'NIL':
        return 'No'
    elif pd.isna(row['dismissal']):
        return 'No'
    else:
        return 'Unknown'



log_raw['dismissal'] = log_raw.apply(readableLog, axis=1)
log_raw['terminate'] = log_raw.apply(set_terminate, axis=1)
```

Code 2: Make dismissal readable

amount based on the activity type. Originally, the values are entered into the columns according to the activity. The goal of this function is to make the amount column a column that maintains consistency with the value match by keeping the value in the specific columns to identify the associated activity.

- The *replace_nan_with_zero* replaces NaN values with zero value in specific columns of the DataFrame. It is useful to ensure that there are no missing values in critical columns that could affect the dataset analysis.

3.2 Dismissal clarity

- The *readableLog* function translates the values in the dismissal column into more readable labels that represent the status of the traffic infraction more clearly. Text values of multiple characters were used to highlight the values entered by this method and differentiate them from the labels already present. After the function application the possible values are ['NP', 'MV', 'Pr', 'Jd', 'N', 'K', '5', '3', 'A', 'I', 'D', 'T', 'E', '@', 'M', 'Q', 'F', 'V', 'U', 'C', 'B', '$', 'Z', 'J', 'R', '2', '4'][4].

[4]Most of this values are meaningless due to the absence of sufficient documentation.

```
log_raw = log_raw.drop(columns=['matricola'])
```

Code 3: Drop matricola

```
log_raw = log_raw[ log_raw['terminate'] != 'Unknown']
```

Code 4: Filter with terminate column

- The *set_terminate* function determines whether the traffic infraction case can be considered terminated, based on the value of the dismissal column, which is then used later to add a column to store this value.

3.3 Delete matricola colum

The matricola column contains only Nan or 0 values; the value 0 is present only at the 'Appeal to Judge' activity while for all other activities the value remains Nan. The column is found to be unwise and therefore removed.

3.4 Observation and notes

The org:resource, points, vehicleClass, article columns have values only when concept:name is Create Fine for all other activities the values are Nan, so these values are obtainable by taking the Create Fine activity. For the notification-Type and lastSent columns, no values were included because they have meaning only for specific activities and therefore kept as Nan values for activities where they are not intended to be used.

4 Data filtering

4.1 Concrete cases

To analyze cases where it is certain that the activity is terminated, cases marked by the previously entered terminated column as unknown were filtered out, ensuring that only cases with a clearly defined status are considered in the analysis.

4.2 Time filtering

To ensure that the analysis focuses on meaningful data, a filter was applied for cases where the duration is not between the minimum other than zero and the maximum among the possible durations.

```
case_durations = pm4py.get_all_case_durations(log_raw)
minimo_raw = min(case_durations)
massimo_raw = max(case_durations)

# Trovato min = 0 -> filtro
minNotZero = lambda  ls : min([x for x in ls if x != 0])
minimo_raw = minNotZero(case_durations)


max_case_duration = max(case_durations)
filtered_log = pm4py.filter_case_performance(filtered_log,
    minimo_raw, max_case_duration,
                                                timestamp_key='
                                                    time:timestamp
                                                    ',
                                                case_id_key='case:
                                                    concept:name'
                                                    )
```

Code 5: Filtering logs for case duration

```
    filtered_log = pm4py.filter_end_activities(filtered_log, ['
        Payment', 'Send for Credit Collection', 'Send Appeal to
        Prefecture', 'Appeal to Judge'])
```

Code 6: Filtering logs for valid end activity

4.3 Start/End activity

Un ulteriore filtro è stato applicato per eliminare i casi in cui le attività iniziali e/o finali non sono valide. Per ogni caso l'attività iniziale è 'Create fine' e il log ha già questa caratteristica. Per l'attività finali le attività valide sono 'Payment', 'Send for Credit Collection', 'Send Appeal to Prefecture', 'Appeal to Judge'.

5 Knowledge uplift trail

This section discusses the process used to analyze the logs. Initially, the log was provided, which was then cleaned and filtered by removing unnecessary and/or incomplete cases for the purpose of analysis. Some statistical methods were applied to the obtained log for preliminary analysis and then it will be divided into two (cases ending in payment and cases not ending in payment) and analyzed and compared. During the analysis, ideas will be developed to improve the service which will be covered in a specific section. During the process, process mining and process discover techniques are used to analyze the actions.

6 Statistical analysis

The goal of this section is to use statistical analysis to analyze logs. Initially, the log in general will be analyzed and then the cases that end in payment will be divided from those that end in the other valid final activities.

6.1 General analysis

Wanting to improve the service we start by analyzing the durations of all cases within the datalog. The following table 6.1 and the graphs (fig. 1) shows some important statistical indices. The calculated statistics provide valuable

Duration	Seconds	
Mean	30599467.08	∼353 days
Max	377740800.0	∼12 year
Min	86400.0	1 day
Std	30000197.11	∼347 days
Quantiles		
0.25	1900800.0	∼22 days
0.50	19526400.0	∼226 days
0.75	52876800.0	∼612 days

Table 1: Statistical index of cases duration

information about the durations of cases in your log. Here are some observations:

Initial Observations

The average duration of cases is about 30,599,467 seconds, which corresponds to about 353 days. This means that, on average, cases take almost a year to complete. However, the maximum duration of cases can reach 377,740,800 seconds, which is equivalent to about 12 years, indicating that some cases may take an extremely long time to be resolved. In contrast, the minimum duration is only 86,400 seconds, equivalent to one day, showing that there are cases that are completed very quickly.

The standard deviation is very high, about 30,000,197 seconds (about 347 days), suggesting great variability in the duration of cases. Analyzing the quantiles, we see that 25% of the cases have a duration of less than 1,900,800 seconds (about 22 days), while the median value, representing 50% of the cases, is 19,526,400 seconds (about 226 days). Finally, 75% of the cases have a duration of less than 52,876,800 seconds (about 612 days). These data show a very wide distribution of case completion times.

We display the durations grouped according to the following groups: Less than one day, less than one week, less than two weeks, less than one month, less than two months, less than three months, less than six months, less than 12 months, less than 18 months, less than 24 months, less than five years, over. The absolute frequency table 6.1 and some graphs (fig. 2) are given.

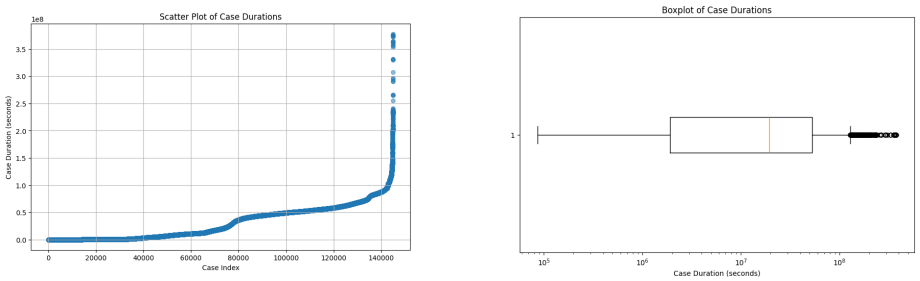

Figure 1: Graphs of durations scatterplot, on the left, and boxplot, on the right.

category	count
Less than one day	10431
Less than one week	23099
Less than two weeks	5182
Less than one month	3995
Less than two months	6803
Less than three months	6070
Less than six months	17102
Less than 12 months	9937
Less than 18 months	11194
Less than 24 months	30936
Less than five years	25378
More than five years	243

From fig. 2 we can observe the following things:

- Distribution of Durations: Most cases have relatively short durations, with a few exceptions extending beyond five years.
- Variability of Durations: The variability of durations increases with the length of the time categories, as evidenced by the box plots.
- Outliers: Outliers are present in many categories, indicating that there are cases with durations significantly different from the average.

6.2 Paid and unpaid cases

Let us consider the activities that end with a payment and compare them with those that end with other activities. For simplicity the resulting log will also be called logPaid while the other part logUnPaid to differentiate it.With this section we want to analyze and find points of possible for improvement. The log are genereted by filtering the general log by the End activity.

The table 6.2 above shows the mean values and standard deviations of case durations for two data sets: logPaid and logUnPaid. These data are crucial for

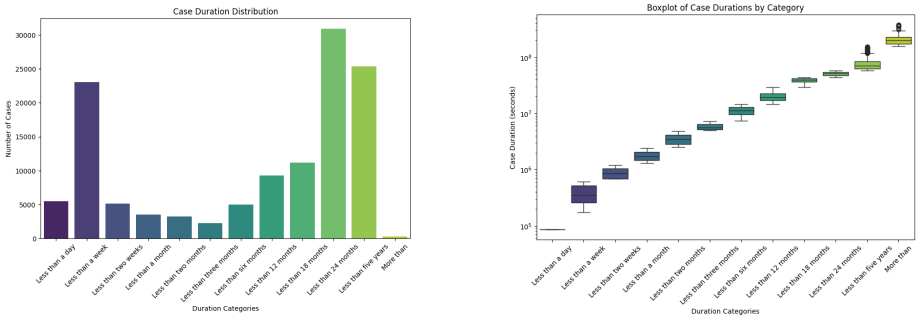

Figure 2: On the right a histogram for grouped durations and on the left a boxplot with the same groupings.

```
logPaid = pm4py.filter_end_activities(filtered_log, ['
    Payment'] )

att_final = ['Send for Credit Collection', 'Send Appeal to
    Prefecture', 'Appeal to Judge']
logUnPaid = pm4py.filter_end_activities(filtered_log,
    att_final)
```

Code 7: Filtering logs for Paid and UnPaid activities

understanding the variability and behavior of cases that terminate successfully (logPaid) versus those that do not (logUnPaid).

- For the logPaid group:
  - Average Duration: The average duration of paid cases is approximately 15 weeks, 2 days, 7 hours, 16 minutes and 41 seconds. This value indicates the average time it takes to complete cases that are successful.
  - Standard Deviation: The standard deviation is about 26 weeks, 4 days, 7 minutes, and 5 seconds, suggesting considerable variability in case durations. A high standard deviation indicates that paid case completion times vary widely, from significantly shorter to much longer durations.
- For the logUnPaid group:
  - Average Duration: The average duration of unpaid cases is approximately 3 years, 6 weeks, 4 days, 12 hours, 46 minutes, and 7 seconds.This value represents the average time to the point where the case failed, suggesting that unpaid cases tend to take much longer than paid cases.

logPaid	
mean	15w:2g:7h:16m:41s
std	26w:4g:0h:7m:5s
logUnPaid	
mean	3y:6w:4g:12h:46m:7s
std	1y:5w:2g:19h:4m:43s

  - Standard Deviation: The standard deviation is approximately 1 year, 5 weeks, 2 days, 19 hours, 4 minutes, and 43 seconds. Again, a high standard deviation indicates significant variability in the completion time of unpaid cases, with some cases possibly taking significantly longer than the average.

- Conclusion: The data suggest that paid cases tend to be completed in a much shorter time interval than unpaid cases, with greater variability in completion time for paid cases. This difference could indicate that issues leading to an unpaid case take longer to resolve or that these cases go through more steps in the process that slow their completion.

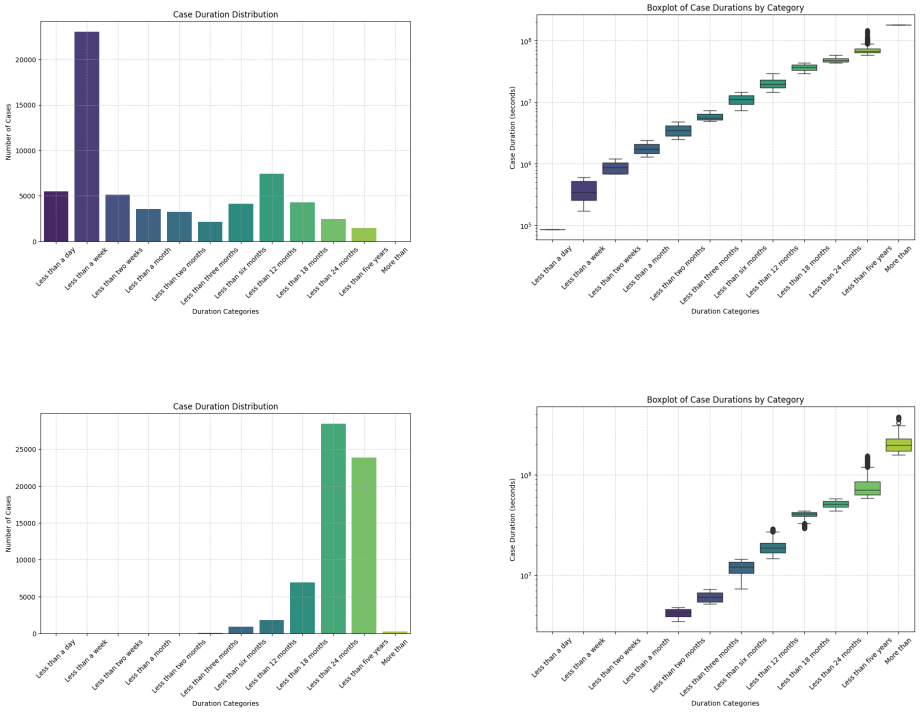

Figure 3: Durations comparison between logPaid(left) and logUnPaid(right)

In the following sections we try to identify the cause of this.

6.2.1 Paid case: logPaid

We analyzed the log of cases completed with payment and derived the mean and standard deviation of task duration. Now we consider the mean duration for each activity.

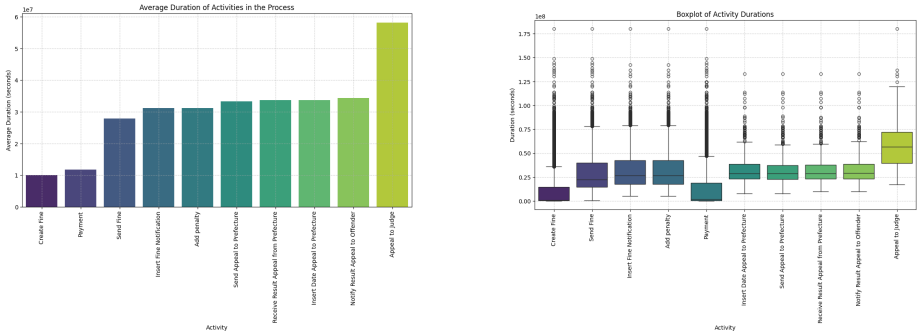

Figure 4: Activities duration graphs for logPaid

This bar chart (fig.6.2.1) presents the average duration of each activity within the process.

- Create Fine and Payment: These activities have the shortest average durations, indicating they are relatively quick to complete.
- Appeal to Judge: This activity stands out with the highest average duration, reinforcing the observations from the boxplot regarding its time-consuming nature.
- Intermediate Activities: Activities like Send Fine, Insert Fine Notification, and Add Penalty have moderate average durations, indicating room for process optimization.
- Efficiency Indicators: The chart provides a clear overview of which activities are most time-consuming, highlighting areas for potential improvement.

This heatmap (fig. 5) illustrates the average waiting times between consecutive activities. The color intensity indicates the length of the waiting time, with darker colors representing longer durations.

- Payment to Add Penalty: This transition shows a significant average waiting time (1.2e+07 seconds), indicating a bottleneck or delay in this step.
- Insert Date Appeal to Prefecture to Receive Result Appeal from Prefecture: This sequence also shows a long waiting time, suggesting delays in processing appeals.

Figure 5: Average waiting time between activities for logPaid

- Create Fine and Send Fine: These activities have relatively shorter average waiting times, indicating a more streamlined process.
- Critical Transitions: The heatmap reveals critical areas where process improvements are necessary to reduce waiting times and enhance efficiency.

This boxplot (fig. 6.2.1) visualizes the distribution of time durations for various activities. Each box represents the interquartile range (IQR) of durations, with the whiskers extending to 1.5 times the IQR from the quartiles. Outliers are represented as individual points. From this we can see:

- Create Fine: This activity has a relatively lower median duration compared to others, indicating it typically takes less time.
- Appeal to Judge: This activity has the highest median duration and a significant number of outliers, suggesting it often takes the longest time and has high variability.
- Send Fine, Insert Fine Notification, and Add Penalty: These activities show a moderate median duration with considerable variability, indicating a consistent yet somewhat unpredictable process.
- General Observations: The variability and presence of outliers in each activity highlight the inconsistency and potential inefficiencies within the process.

6.2.2 Paid case: logUnPaid

Now let's do the same analysis with logUnPaid.

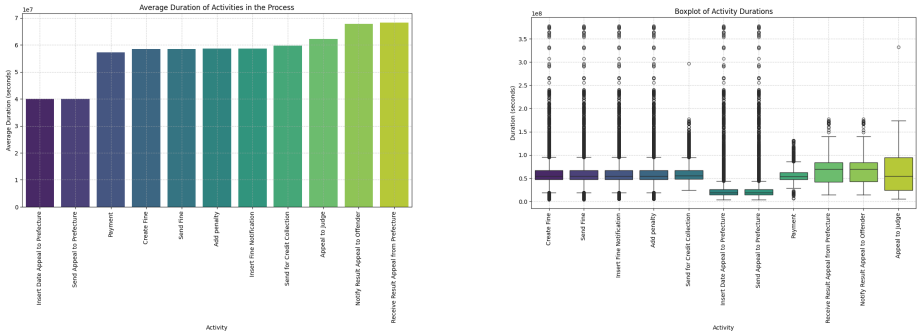

Figure 6: Activities duration graphs for logUnPaid

From this bar chart (fig. 6.2.2) we can see:

- Create Fine and Payment: These activities have the lowest average durations, indicating that they are relatively quick to complete.
- Appeal to Judge: This activity stands out as having the highest average durations, confirming the boxplot's observations about its time-consuming nature.
- Intermediate Activities: Activities such as Send Fine, Insert Fine Notification, and Add Penalty have moderate average durations, indicating room for process optimization.

Efficiency Indicators:

The graphs provide a clear overview of the most time-consuming activities, highlighting areas with the greatest potential for improvement.

From this heatmap (fig. 6.2.2) we can see:

- "Insert Fine Notification" to "Notify Result Appeal to Offender": This transition has the shortest average wait time (0 seconds), indicating a fast process.
- Transitions Involving "Appeal to Judge": Tend to have longer wait times, especially when moving to or from "Notify Result Appeal to Offender," suggesting inefficiencies in these areas.
- Long Wait Times: Some combinations of activities, such as from "Receive Result Appeal from Prefecture" to "Insert Date Appeal to Prefecture," show significantly long wait times.

Bottleneck Indicators: The heatmap highlights critical areas where wait times are longer, indicating points in the process that need optimization.

From this boxplot (fig. 6.2.2) we can see:

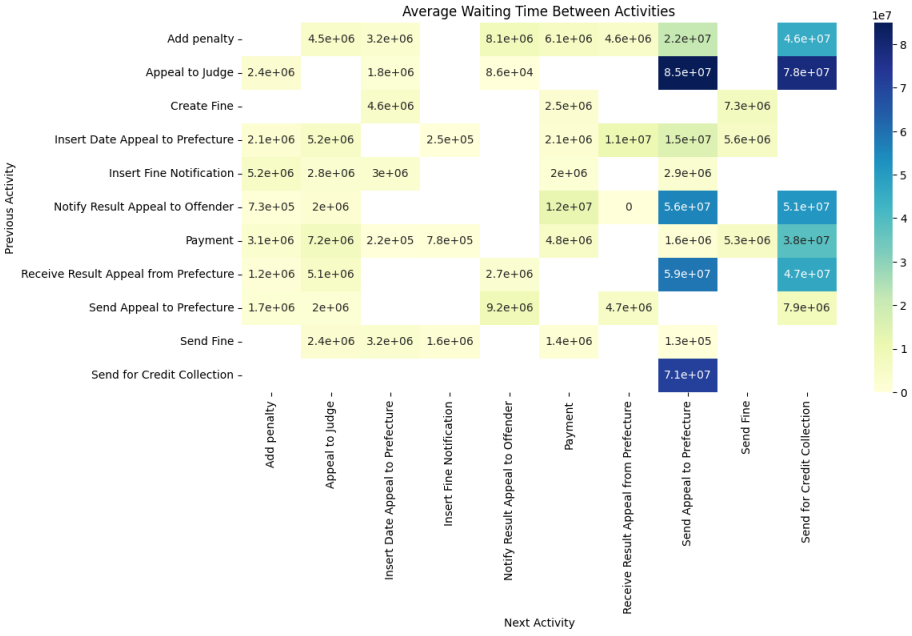

Figure 7: Average waiting time between activities for logUnPaid

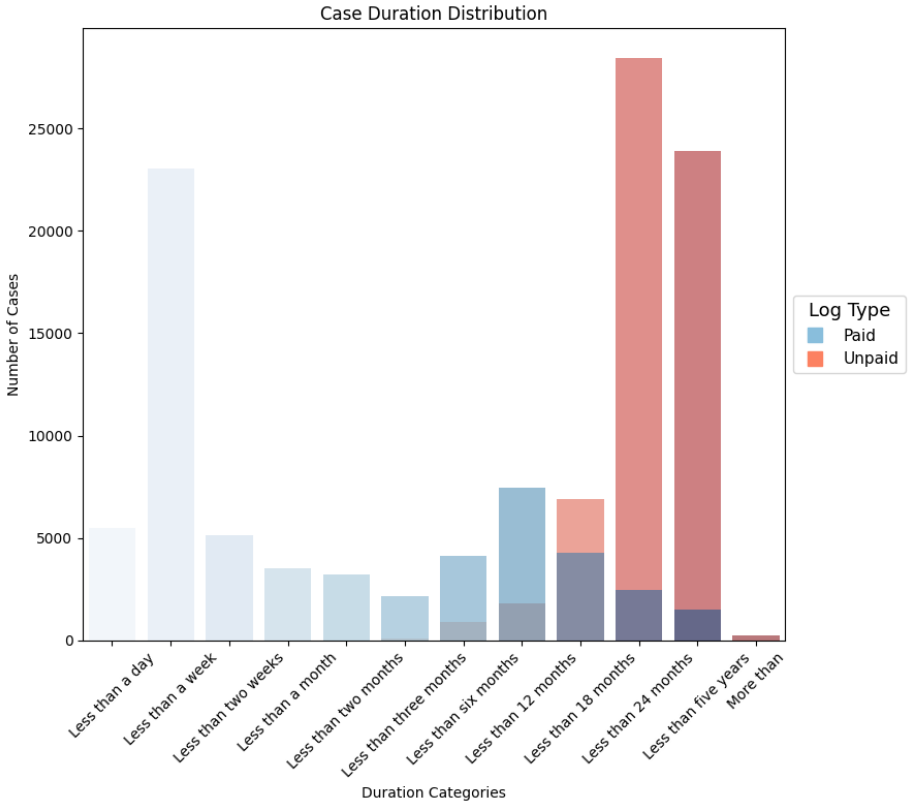

Figure 8: Comparative histogram between durations of logPaid and logUnPaid

- Create Fine: This activity has a relatively low median duration compared to the other activities, suggesting that it generally takes less time.
- Send Fine, Insert Fine Notification, and Add Penalty: These activities show moderate median durations with considerable variability, indicating a consistent but sometimes unpredictable process.
- Appeal to Judge: This activity has the highest median duration and a significant number of outliers, suggesting that it often takes longer and has high variability.
-

Efficency indicator: The presence of numerous outliers and high variability in durations indicate inefficiencies in the process that could be improved to achieve greater consistency.

Finally, the histograms of the two logs are directly superimposed. The graph shows, on a scale of blues, the durations in the paid log and it can be seen that

most cases are resolved within the first week while, on a scale of reds, the unpaid logs show a different situation in which most are resolved within 24 months.

6.2.3 Observation

- The boxplots of the durations of the activities provide a visual representation of the time distributions of the different stages of the process. Analyzing the boxplot, one can observe the presence of numerous outliers suggests that there are exceptionally long instances for each activity, highlighting possible inefficiencies or variability in completion times. In particular, the "Appeal to Judge" and "Notify Result Appeal to Offender" activities show greater variability, as evidenced by the length of the boxes and the number of outliers. The "Create Fine" activity has a relatively low median of durations, suggesting that it generally takes less time to complete, while "Appeal to Judge" has the highest median and significant variability, indicating that it is one of the longest and most unpredictable activities. These observations highlight inconsistency in the process and suggest the need for interventions to standardize procedures and reduce time variations, thereby improving the overall efficiency of the process.

- Heatmaps provide a clear visualization of average wait times between process steps. This analysis shows that the transition from "Insert Fine Notification" to "Notify Result Appeal to Offender" has the shortest average wait time, with 0 seconds, suggesting a very rapid transition between these activities. In contrast, transitions involving "Appeal to Judge" tend to have significantly longer wait times, indicating possible bottlenecks in the process. In particular, the transition from "Receive Result Appeal from Prefecture" to "Insert Date Appeal to Prefecture" has a significantly high average wait time, signaling a potential bottleneck that needs action to improve efficiency. These observations suggest that activities related to appeals, particularly those involving the judge, could benefit from an overhaul to reduce wait times and optimize the flow of the process.

- Bar graphs of the average durations of activities in the process provide a clear indication of how long, on average, each activity takes. The "Create Fine" and "Payment" activities appear to have the lowest average durations, indicating that they are relatively quick to complete. In contrast, the "Appeal to Judge" and "Notify Result Appeal to Offender" activities stand out as having the highest average durations, reinforcing what was observed in the other analyses regarding their complexity and variability. Most of the activities fall in an average duration range between $2 * 10^7$(230 days approximately) and $4 * 10^7$ (460 days approximately) seconds, suggesting some consistency in completion timelines, but also room for further optimization. Identifying activities with high average durations allows improvement efforts to be focused on these points in order to reduce overall process times and increase operational efficiency.

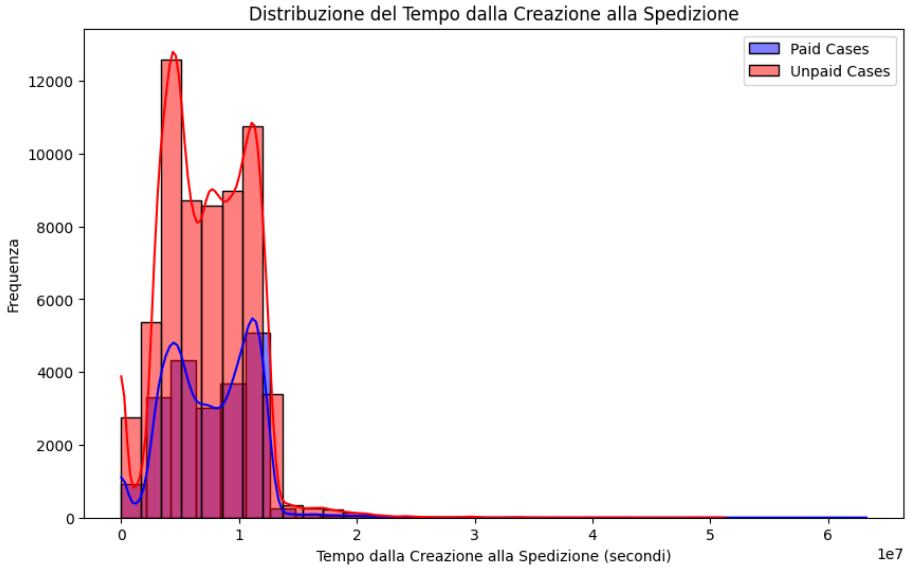

Figure 9: Comparison time Create-Send for logPaid and logUnPaid

Analysis of the three graphs reveals a clear picture of activity durations and waiting times in the process. The significant variability and the presence of numerous outliers in activity durations suggest that there is room to standardize and optimize procedures, thereby reducing inefficiencies. Waiting time heatmaps highlight critical transitions, especially those related to appeals, which represent potential bottlenecks and areas where action is needed to improve process flow. Finally, average activity durations clearly indicate which stages are the most time-consuming, such as "Appeal to Judge," and which are relatively quick, such as "Create Fine" and "Payment." These observations provide a solid basis for identifying areas for improvement and proposing targeted interventions to optimize overall process efficiency.

Time beetween Create and Send

Assuming that some activities have an excessively long duration due to bureaucracy we consider only those the time elapsed between 'create end' and 'send end'[5] and assume that by improving communication it is possible for a new case to end up in logPaid. The test used is the t-test for independent samples is used to determine whether there are significant differences between the means of two independent groups. The test result are:

- T-statistic: 9.928369333950219
- P-value: 3.3586386693449295e-23

[5]this two activity are the common ones for logPaid cases and logUnPaid ceses

The T-statistic value as high as 9.928 indicates a large difference between the average creation and dispatch times for paid and unpaid cases, the P-value of 3.358e-23, much less than 0.05, suggests that the probability of observing such a large difference between the groups by pure chance is extremely low. This high level of significance allows us to reject the hypothesis that there is no difference between the groups with high confidence. From this we can say that the difference between the mean creation and mailing times for paid and unpaid cases is highly significant, indicating that the timing between the creation and mailing of the fine has a significant effect on the payment outcome. These results support the hypothesis that the time from fine creation to dispatch is a critical factor in payment success. Paid cases tend to have a different time interval between these two activities than unpaid cases.

These analyses lead to these considerations:

- Optimizing the process of sending fines by reducing the time between creation and mailing is recommended. Establishing procedures or standards to ensure that fines are sent as quickly as possible after they are created could be helpful.
- Implement a monitoring system to continuously track and analyze these times. This would help to quickly identify any delays and take immediate corrective action.

```
    top5_logPaid = pm4py.filter_variants_top_k(logPaid, 5)
net, im, fm = pm4py.discover_petri_net_inductive(top5_logPaid,
    noise_threshold=.4)
pm4py.view_petri_net(net,im, fm, format='png')
```

Code 8: Build petri net with the most frequent variant

7 Process discover

7.1 Paid cases

The following table summarizes the five most frequent observed variants for paid cases:

Variants	Count	%
Create Fine, Payment	46371	0.69
Create Fine, Send Fine, Insert Fine Notification, Add penalty, Payment	9518	0.14
Create Fine, Send Fine, Insert Fine Notification, Add penalty, Payment, Payment	3735	0.06
Create Fine, Send Fine, Insert Fine Notification, Payment, Add penalty, Payment	3301	0.05
Create Fine, Send Fine, Payment	3131	0.05

Table 2: Top five variants of logPaid.

The goal of this section is to provide a global representations of these variants and how the activities are related one each other.

The Petri net of the five most frequent variants of the logPaid log is shown below.

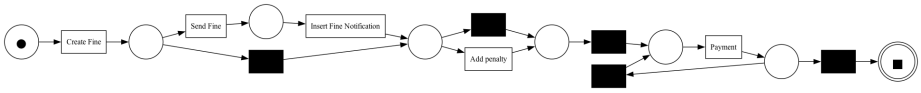

Figure 10: Petri net of the 5 most frequent variant

Figure 11 shows the petri net of the best variant. As shown after creation, the payment activity is present. Figure 12 shows the case where a payment notification is sent, as can be seen the sequence of notification and exclusive and end with payment, in the second case the payment also includes the penalty entered by the activity. The last petri net (fig. 13) was inserted to emphasize that the payment activity can be done multiple times.

7.2 UnPaid cases

The following table summarizes the five most frequent observed variants for unpaid cases[6]:

[6]To compact the table the second row represents the initial actions common to all subsequent variants, they do not represent a variant.

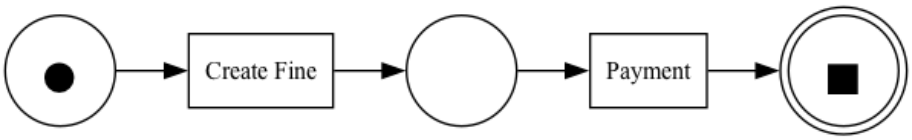

Figure 11: Simplest variant

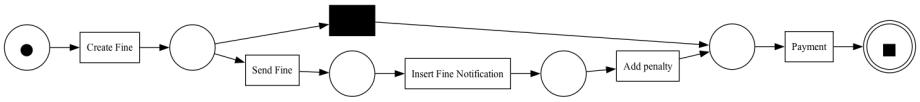

Figure 12: Added penality

Variants	Count	%
Create Fine, Send Fine, Insert Fine Notification		
..,Add penalty, Send for Credit Collection	56473	0.91%
.., Insert Date Appeal to Prefecture, Add penalty, Send Appeal to Prefecture	2494	0.04%
.. Add penalty, Payment, Send for Credit Collection	1514	0.02%
.., Payment, Add penalty, Send for Credit Collection	522	0.01%
.., Add penalty, Insert Date Appeal to Prefecture, Send Appeal to Prefecture	442	0.01%

Table 3: Top five variant of logUnPaid.

The goal of this section is to provide a global representations of these variants and how the activities are related one each other. The Petri net of the five most frequent variants of the logunPaid log is shown below(the code is the same of the previous).

Figure 14 shows the petri net of the best variant. After creating the fine, the sequence of activities continues with notification, adding a penalty, and concludes with 'send to credit collection'. The next image (fig. 15) shows the case where you have the option to appeal to the prefecture and finally fig. 17 is shown that there is a payment activity but it is not terminal.

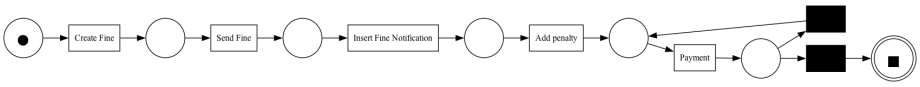

Figure 13: Variant with multiple Payment action

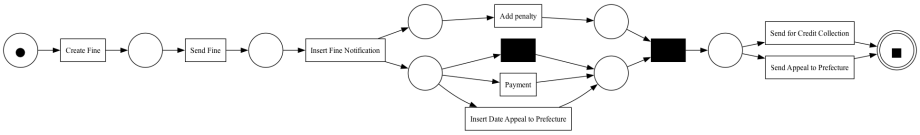

Figure 14: Petri net of the 5 most frequent variant for the unpaid log

8 Conformance checking

A comprehensive process model has been generated using the inductive miner algorithm, integrating the local workflows of both logs, logPaid and logUnPaid. This model provides a complete and generalized view of possible scenarios during process execution, capturing a wide range of potential situations and offering a robust overview of the entire process.

Table 8 provides a detailed overview of the distribution of fitness values among the specified ranges. Analyzing the data, it is observed that the majority of instances are in the 0.65-0.7 fitness interval, with a remarkable 80.81% of observations, indicating a predominance of high-performing results. This range represents a clear concentration of positive results, suggesting that the models or processes analyzed are generally very effective.

The fitness intervals with lower values (0.5-0.55 and 0.55-0.6) show very low percentages, 0.10% and 0.74%, respectively, suggesting that lower-performing results are relatively rare. In contrast, the intervals above 0.7, except the interval of 0.7-0.75 with a very low percentage (0.03%), show a more significant distribution, with the interval 0.75-0.8 at 14.43% and the interval 0.8-0.85 at 1.09%.

The intervals with fitness values above 0.85 represent a relatively small percentage, but the values are still significant. In particular, the range 0.85-0.9, although modest with 2.52%, and the range 0.9-0.95 and 0.95-1.0 with very low

```
    combined_log = pd.concat([top5_logUnPaid, top5_logPaid],
        ignore_index=True)

net, im, fm = pm4py.discover_petri_net_inductive(combined_log,
    noise_threshold=.3)
pm4py.view_petri_net(net,im, fm, format='png')
```

Code 9: Build petri net

Figure 15: Simplest variant

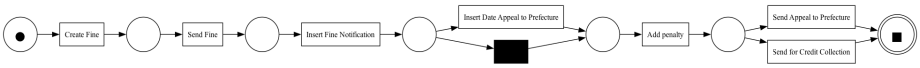

Figure 16: Added penality

percentages (0.03% and 0.00% respectively), indicate that very few instances reach the highest fitness values.

In summary, the distribution of fitness values shows a predominant trend toward high values, highlighting the general effectiveness of the models or processes under consideration. However, the presence of a few instances with very high fitness values suggests that there may be room for further optimization or more in-depth analysis of exceptions.

9 Organizational goal

The organizational objectives diagram outlines an approach to business process optimization. The goals to be pursued are mainly two: to improve customer satisfaction and to reduce shipping time. The first goal can be divided the two separate points improve communication (company - customer) and speed up some processes (customer - company). To increase user satisfaction and improve the efficiency of the fine management system, we will work with technology service providers to develop and implement an automated notification system. This system will have the dual purpose of optimizing communication time, eliminating unnecessary costs, and simplifying the management of fines. This will both reduce dispatch time but also allow for quick communication directly with

Fitness Bin	Count	Percentage
0.5-0.55	24	0.10%
0.55-0.6	186	0.74%
0.6-0.65	64	0.25%
0.65-0.7	20,391	80.81%
0.7-0.75	7	0.03%
0.75-0.8	3,640	14.43%
0.8-0.85	276	1.09%
0.85-0.9	635	2.52%
0.9-0.95	8	0.03%
0.95-1.0	1	0.00%

Table 4: Table for fitness values distribution

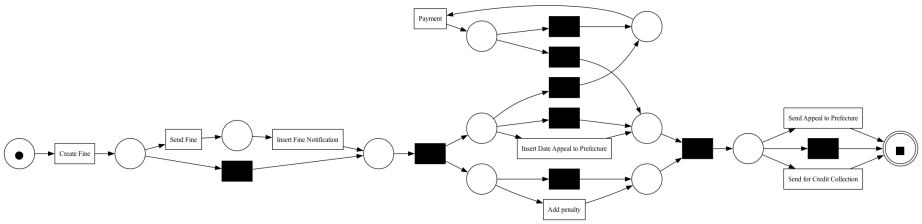

Figure 17: petri net obtained from the union of the 5 most frequent variants of logPaid and logUnPaid

the customer. The same system can be used for communication in the opposite direction (e.g., for disputing fines). The organizational objectives diagram outlines an approach to business process optimization. The goals to be pursued are mainly two: to improve customer satisfaction and to reduce shipping time. The first goal can be divided the two separate points improve communication (company - customer) and speed up some processes (customer - company). To increase user satisfaction and improve the efficiency of the fine management system, we will work with technology service providers to develop and implement an automated notification system. This system will have the dual purpose of optimizing communication time, eliminating unnecessary costs, and simplifying the management of fines. This will both reduce dispatch time but also allow for quick communication directly with the customer. The same system can be used for communication in the opposite direction (e.g., for disputing fines). The organizational objectives diagram outlines an approach to business process optimization. The goals to be pursued are mainly two: to improve customer satisfaction and to reduce shipping time. The first goal can be divided the two separate points improve communication (company - customer) and speed up some processes (customer - company). To increase user satisfaction and improve the efficiency of the fine management system, we will work with technology service providers to develop and implement an automated notification system. This system will have the dual purpose of optimizing communication time, eliminating unnecessary costs, and simplifying the management of fines. This will both reduce dispatch time but also allow for quick communication directly with the customer. The same system can be used for communication in the opposite direction (e.g., for disputing fines).

10 Futher work

For further analysis, one might consider further splitting the two logs before and after 18 months to analyze and understand the relationship between penalties and case actions. One might also consider optimizing the terminal nonpayment actions by establishing a policy that improves the bureaucratic efficiency of the company. To further improve this analysis, additional information should

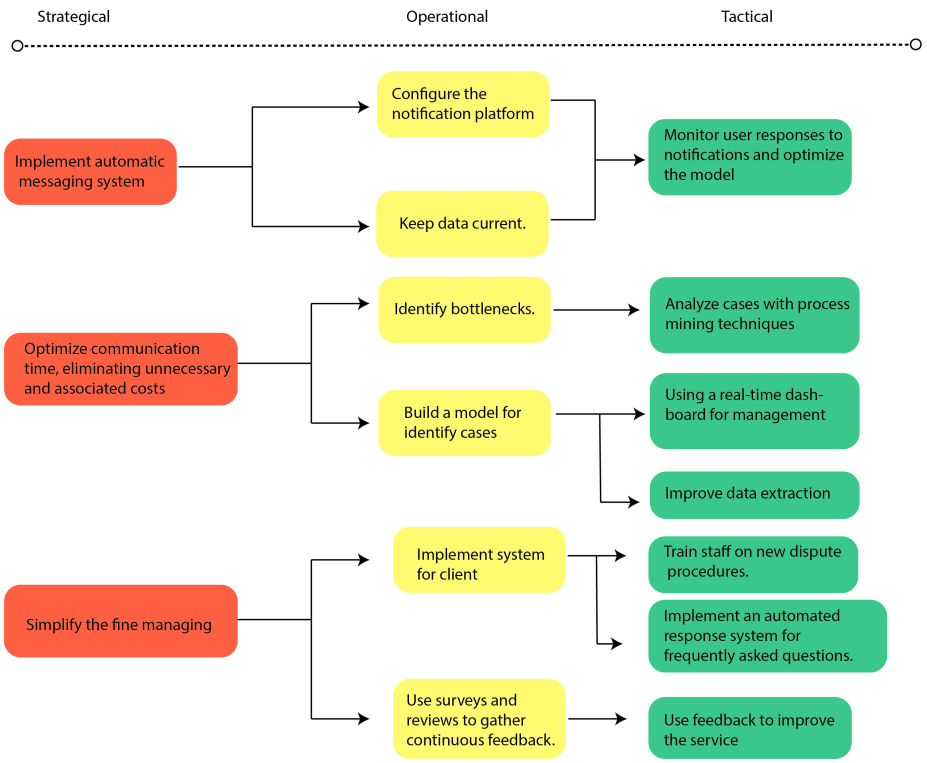

Figure 18: Business goals

be collected, such as geographic area, area classification (urban city, suburbs, highways), or gender of the offender. By adding these data, we can perform clustering analysis to identify patterns and trends within the dataset. This, in turn, would allow us to build more accurate predictive models that rank the probability of paying fines based on several independent variables. In conclusion, the solution can be developed through machine learning models to help staff by getting more effective results and use a system for better communication.

11 Reference

- **Data**: Files for the data were made available through Ariel site.
- **Code**: The jupiter notebook used is available on GitHub