LOAN APPLICATION

Ikram Bourras

Business Information Systems
Prof. Paolo Ceravolo
a.a. 2024 - 2025

Contents

1. Introduction

This project aims to conduct an in-depth analysis of the loan application management process within a Dutch financial institution. Utilizing a real event log from the BPI Challenge 2012, which comprises 13,087 cases and 262,200 events, we will delve into the operational dynamics to identify inefficiencies and opportunities for improvement. The analysis will focus on three interconnected subprocesses: **application management** (from submission to final decision), **offer management** (from preparation to dispatch), and **manual activities related to work items**, such as anti-fraud checks and follow-ups. Particular attention will be paid to the **AMOUNT_REQ** (requested amount) attribute, a key factor that could significantly influence processing times and final outcomes. The objective is to detect the numerous decision points and manual activities that characterize the process, identifying potential bottlenecks to optimize overall efficiency.

2. Dataset Description and Organization

2.1 Log Length

The log details events related to personal loan and overdraft requests, recorded over a period of approximately six months: from October 1, 2011, to March 14, 2012.

```
#periodo del log

import pandas as pd
# convert to datetime
df['time:timestamp'] = pd.to_datetime(df['time:timestamp'])
# find min and max date
min_date = df['time:timestamp'].min()
max_date = df['time:timestamp'].max()
# print difference
print(f"Min date: {min_date}")
print(f"Max date: {max_date}")
print(f"Difference: {max_date - min_date}")
```

```
Min date: 2011-10-01 00:38:44.546000
Max date: 2012-03-14 16:04:54.681000
Difference: 165 days 15:26:10.135000
```

2.2 Data Volume

- 13,087 cases
- 262,200 total events
- **24 unique event classes** at the start of the analysis

```
# prompt: numero dei casi, eventi totali e numero classi di eventi uniche

print("Numero di casi:", df['case:concept:name'].nunique())
print("Eventi totali:", len(df))
print("Numero di classi di eventi uniche:", df['concept:name'].nunique())
```

```
Numero di casi: 13087
Eventi totali: 262200
Numero di classi di eventi uniche: 24
```

2.3 Data Attributes

The dataset contains both **case-level** and **event-level attributes**.

2.3.1 CASE-LEVEL ATTRIBUTES

- **AMOUNT_REQ**: The amount requested by the customer.
- **case:reg_date**: The registration date of the application.
- **case:concept:name**: The unique identifier for the case (case ID).

2.3.2 EVENT-LEVEL ATTRIBUTES

- **time:timestamp**: The exact date and time the event was completed.
- **concept:name**: The name of the activity or process phase.
- **lifecycle:transition**: The lifecycle phase of the activity (e.g., Schedule, Start, Complete).
- **org:resource**: The resource (person or system) responsible for the event.

2.3.3 KEY LOG ATTRIBUTES

org:resource

This field identifies the person, system, or department that performs an activity within the process. There are **68 unique resources**.

```
# prompt: org:resource valori unici

print("\nValori unici per 'org:resource':", df['org:resource'].unique())
print("Numero di valori unici per 'org:resource':", df['org:resource'].nunique())
```

```
Valori unici per 'org:resource': ['112' nan '10862' '10913' '11049' '10629' '11120' '10809' '10912' '11201'
 '11119' '10861' '11203' '11181' '11189' '10609' '11111' '10982' '11019'
 '11180' '10899' '10138' '11002' '11122' '10889' '10972' '11121' '10939'
 '11029' '11009' '11000' '10863' '11169' '11179' '11001' '10971' '10228'
 '11202' '10789' '10881' '10909' '10188' '10910' '10929' '10931' '11259'
 '11200' '10779' '10880' '10914' '10859' '11339' '10933' '11079' '10932'
 '10935' '11254' '11003' '10125' '11269' '10821' '11289' '10124' '11299'
 '11309' '11300' '11302' '11319' '11304']
Numero di valori unici per 'org:resource': 68
```

Questa informazione è **fondamentale** per l'analisi organizzativa perche si puo andare a capire Distribuzione del carico di lavoro → capire chi lavora di più.

```
Statistics for 'org:resource':
Max count: 45687
Resource(s) with max count:
['112']

Min count: 2
Resource(s) with min count:
['10821']

Median count: 2413.5
Mode count(s):
[6]
Resource(s) with mode count(s):
['10125', '11269']
```

The workload among the 68 resources varies significantly, ranging from 2 to nearly 46,000 events. The resource with the highest workload is an automated system ('112'). The **median** (2413.5) indicates that half of the resources have a workload below approximately 2400 events, while the **mode** (6) shows that many resources have a low workload. This highlights a strong imbalance between resources with high workloads (like automated systems) and those with low workloads (human or infrequently used).

lifecycle:transition

This attribute indicates the lifecycle phase of the activity.

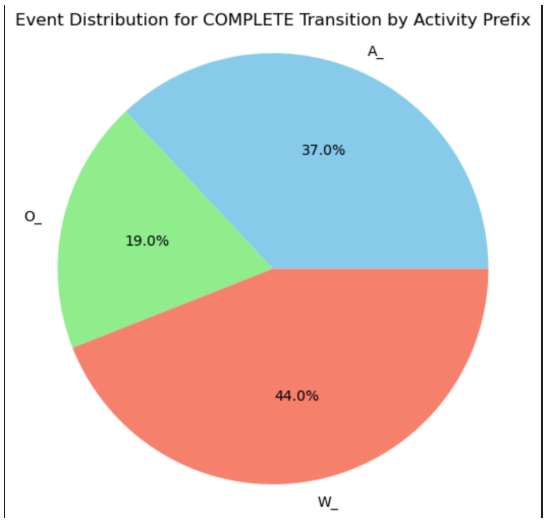
```
[24] #lifecycle:transition valori unici

     print("\nValori unici per 'lifecycle:transition':", df['lifecycle:transition'].unique())
     print("Numero di valori unici per 'lifecycle:transition':", df['lifecycle:transition'].nunique())
```

```
Valori unici per 'lifecycle:transition': ['COMPLETE' 'SCHEDULE' 'START']
Numero di valori unici per 'lifecycle:transition': 3
```

- **COMPLETE**: Indicates the **conclusion of an activity or work item**. It marks the moment when the activity has been finished.
- **SCHEDULE**: Indicates that a work item has been **planned or queued for future execution**.
- START: Indicates the beginning of the execution of an activity or work item.

For activities of type **A_** (Application) and **O_** (Offer), the **complete** event is the only one recorded. For **W_** type activities, the stages are distinguished as: **schedule** (scheduled), **start** (started), and **complete** (completed).



concept:name

Application Events (A_)

These events track the overall status of the application, from the moment the customer submits it to the final outcome:

- **A_SUBMITTED / A_PARTLYSUBMITTED**: Request submitted, either complete or partial.
- **A_PREACCEPTED**: Preliminary acceptance, pending additional information.
- **A_ACCEPTED**: Request accepted, awaiting verification.
- **A_FINALIZED**: Application complete, ready for the final decision.
- **A_APPROVED / A_REGISTERED / A_ACTIVATED**: Application approved and, in some cases, activated.
- **A_CANCELLED**: Request cancelled.
- **A_DECLINED**: Request rejected.

Offer Events (O_)

These events focus on the management of offers to the customer:

- O_SELECTED / O_PREPARED / O_SENT / O_SENT BACK / O_ACCEPTED: Various phases of the offer.
- **O_CANCELLED / O_DECLINED**: Cancellation or rejection of the offer.

Workflow Events (W_)

These events pertain to manual activities and verifications performed by personnel:

- **W_Afhandelen leads**: Management of initial requests.
- **W_Completeren aanvraag**: Completion of information.
- **W_Nabellen offertes**: Follow-up after sending offers.
- **W_Valideren aanvraag**: Final validation of the request.
- **W_Nabellen incomplete dossiers**: Retrieval of missing information.
- W_Beoordelen fraude: Fraud control.
- **W_Wijzigen contractgegevens**: Modification of contract data.

time:timestamp

This records the **date and time of the event**. It's essential for ordering events, calculating performance, and analyzing bottlenecks and execution times.

case:REG_DATE

This indicates the **registration date of the application** and allows for analyzing how performance changes over time.

case:concept:name

This is the **unique case identifier (case ID)**, which links all events of the same request and enables the formation of traces in Process Mining.

case:AMOUNT_REQ

This is the **amount of the requested loan**. It's useful for segmenting requests and analyzing the impact of the amount on the outcome or process duratio

3.Organisational Goals

3.1 Understanding Underlying Business Processes

The objective is to gain a detailed view of how the process actually unfolds, based on event data collected in the logs. The main elements of this analysis include:

- **Frequency and distribution of request outcomes:** Calculate and visualize the number and percentage of cases concluded with outcomes such as Approved, Rejected (e.g., final event A_DECLINED), Cancelled (A_CANCELLED, O_CANCELLED), and still open cases. The analysis showed that the A_DECLINED outcome is the most frequent among rejections, while A_APPROVED is never a final event, indicating that closing cases on this event does not reflect operational reality.
- **Average cycle times by outcome:** Calculate the average throughput time for each outcome category, visualizing the time distribution using boxplots or violin plots to identify variations and outliers.
- **Process model mapping and discovery (AS-IS):** Use Process Mining tools like pm4py to generate models based on inductive Petri nets (with an inclusion threshold of 0.9), Directly-Follows Graphs, or Heuristic Miner, in order to accurately represent the actual flow of requests and reduce noise.
- **Process variant analysis:** Count the existing variants (using functions like `get_variants_df` from pm4py), identify the most common variants, and evaluate their frequency, to understand operational differences between cases.
- **Analysis of resources and their activity:** Examine the distribution of activities among the involved resources, checking who performs multitasking and who has a more specialized role, to study the impact on performance.

3.2 Identifying Opportunities for Improved Efficiency and Effectiveness

- **Identification of bottlenecks and slow activities:** Calculate the average times of individual activities and identify critical phases with high execution times that slow down the entire process.
- **Comparative analysis between multitasking and non-multitasking resources:** Compare execution and turnaround times to understand if multitasking positively or negatively affects performance.
- **Conformance checking:** Align real logs with the AS-IS process model to identify significant deviations and evaluate their operational and quality impact.
- **Improvement of data quality and reliability:** Proceed with log cleaning and normalization for a more complete and precise analysis.

4. Definition of Organizational Strategic Objectives (KPIs)

KPIs are quantitative measures that directly link business activities to strategic, operational, and tactical objectives. Their definition derives from process understanding and specific requirements, measuring results against predetermined goals.

4.1 Quantitative KPIs

- **Frequency and distribution of outcomes** (Approved, Rejected, Cancelled, Pending).
- Average cycle times by outcome.
- Number and frequency of process variants.
- Average times and standard deviations of individual activities.
- Percentage of cases that exceed the 30-day time threshold (which can cause automatic cancellation).

4.2 Practical KPIs

- Number of applications processed (submitted, accepted, rejected).
- Number of loan offers sent (O_SENT).
- Percentage of activities performed per resource, with multitasking verification.
- **Identification and localization of bottlenecks** (activities with throughput time above average).
- Performance differences between multitasking and non-multitasking resources.

4.3 Directional KPIs

- Temporal analysis of activity distribution (e.g., dotted chart).
- Evolution over time of approval and rejection rates.
- Percentage of deviations from the AS-IS model (conformance checking).
- Trend of deviation and non-conformity over time.

4.4 Actionable KPIs

- **Mapping of the AS-IS process model** (Directly-Follows Graph, Heuristic Miner).
- **Identification of at-risk activities** (phases with high times or anomalies/outliers).
- Identification of critical variants with worse or better performance.
- Predictive models to estimate success or delay probability.

4.5 Financial KPIs

- Average requested amount per outcome (AMOUNT_REQ).
- Distribution of amounts for approved and rejected cases.

5. Definition of Operational and Tactical Objectives

Organizational success relies on integrating two fundamental forms of knowledge:

- **Descriptive Knowledge**: How things truly are, based on observed data and KPIs that measure current performance.
- **Prescriptive Knowledge**: How things *should* be, indicating necessary actions to achieve objectives, guided by target KPIs and best practices.

5.1 Strategic Objectives

Descriptive Knowledge In the report, I applied data exploration tools (KUT) and descriptive analysis to understand the current state of the organization at a macro level. This included **data cleaning** and selecting relevant data through filters to ensure the quality of the analysis. The goal was to obtain an **accurate snapshot of the context**, identifying trends and patterns in aggregated data, without defining specific actions or targets.

5.2 Tactical Objectives

Descriptive Knowledge At the tactical level, the work involved a **detailed analysis of data subsets** (filtering and segmentation) to highlight specific behaviors or anomalies during certain periods or in particular areas. The objective was to clearly **describe how individual organizational components or processes behave**, based on actual data, to provide a solid foundation for future analyses or decisions.

5.3 Operational Objectives

Descriptive Knowledge At the operational level, I examined **detailed data at a daily or single-transaction level**, using filters and data cleaning techniques to remove noise and outliers. The goal was to **ensure that the analyzed data is accurate and reliable**, to faithfully describe daily activities, without directly intervening with corrective actions or operational changes.

6. Knowledge Uplift Trail

6.1 Data Cleaning

Data cleaning prepares the event log by standardizing data, filling in missing values, or converting categorical variables to numerical ones.

6.1.1 ATTRIBUTE NORMALIZATION

In this specific case, the attribute types are already correct. For instance, we have `date:time` for both `time:timestamp`and `case:Reg_date`.

```
#controllare gli attributi di che tipo sono

print("\nData types of attributes:")
for col in df.columns:
  print(f"- {col}: {df[col].dtype}")
```

```
Data types of attributes:
- org:resource: object
- lifecycle:transition: object
- concept:name: object
- time:timestamp: datetime64[ns]
- case:REG_DATE: datetime64[ns]
- case:concept:name: object
- case:AMOUNT_REQ: object
```

6.1.2 RECONCILIATION OF DUPLICATES AND INCONSISTENCIES

We compared **START** and **COMPLETE** events for all **W_** activities. This allowed us to identify inconsistent activities. We then associated the **COMPLETE** events with their corresponding **START** events, eliminating "orphan" **COMPLETE** events to restore consistency.

```
#Confrontare il numero di eventi START e COMPLETE per ogni attività che inizia con W_

# Filter events starting with 'W_'
df_W = df[df['concept:name'].str.startswith('W_')]

# Count START and COMPLETE transitions for these events
start_W_count = len(df_W[df_W['lifecycle:transition'] == 'START'])
complete_W_count = len(df_W[df_W['lifecycle:transition'] == 'COMPLETE'])

print(f"\nNumero di eventi 'START' per attività che iniziano con 'W_': {start_W_count}")
print(f"Numero di eventi 'COMPLETE' per attività che iniziano con 'W_': {complete_W_count}")
print(f"differenza con 'W_': ", complete_W_count- start_W_count)
```

```
Numero di eventi 'START' per attività che iniziano con 'W_': 71376
Numero di eventi 'COMPLETE' per attività che iniziano con 'W_': 72413
differenza con 'W_':  1037
```

```
print( len(df),"-->",len(df_cleaned))
```

```
262200 --> 261160
```

6.2 Data Filtering

Data filtering sets conditions to filter the data, keeping only cases that meet specific criteria.

6.2.1 FILTERING NOISE

Noise in an event log refers to incorrect, imprecise, or incomplete information.
To address this, I've taken several steps:

- **Ordered the log** by case (`case:concept:name`) and timestamp (`time:timestamp`) to ensure correct chronological sequence.
- For each case, I **extracted the first and second events** (`concept:name`) and calculated their frequency of occurrence.
- I **calculated the duration of each case** as the difference between the last and first timestamp.
- **Merged consecutive `A_SUBMITTED` and `A_PARTLYSUBMITTED` events** into a single `A_SUBMITTED/A_PARTLYSUBMITTED` event, using the timestamp of the second event to reduce redundancy.

My approach offers several advantages:

- **Simplification of analysis**: I reduce the number of distinct states, making the logs easier to read and interpret.
- **Greater focus on relevance**: I concentrate on events that have a tangible impact or measurable duration.
- **Noise reduction**: I eliminate events that could be considered secondary details or "noise" in my main analysis.
- **Improved process representation**: Since `A_PARTLYSUBMITTED` is essentially an echo or temporary confirmation of `A_SUBMITTED`, consolidating them gives me a cleaner, more linear view of the actual flow.

Subsequently, I noticed the presence of two other events ending with "schedule." However, since these events will be filtered in a later stage, I didn't merge them at this point.
Lastly, I considered grouping identical events that repeat in sequence. Nevertheless, I found that the computational effort required for this type of grouping was too high compared to the benefits I'd gain. For this reason, I decided not to proceed with it in this phase.

Managing Complete and Incomplete Cases
I divided the dataset into two parts:

- **Successful cases**: Those ending with the `W_Valideren aanvraag` activity, which indicates the regular conclusion of the case (as per documentation).
- **Rejected cases**: These include final events such as `A_DECLINED`, `A_CANCELLED`, `O_CANCELLED`, and `A_APPROVED`.

I applied an additional filter:

- Only cases ending with `W_Valideren aanvraag` where the last activity is marked as `complete` in the `lifecycle:transition` field are considered valid and retained.

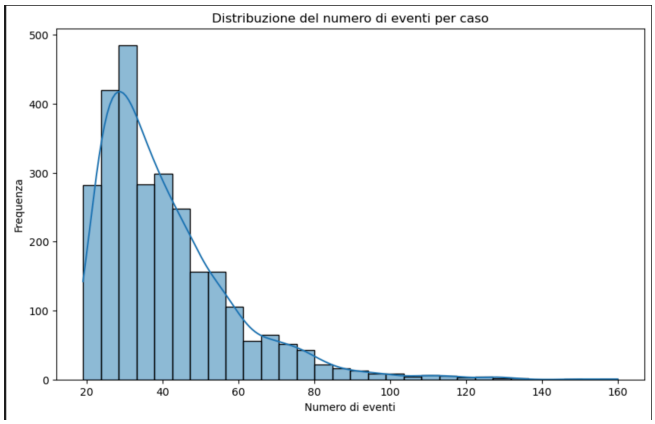
- Any cases without this confirmation are deemed incomplete or irrelevant and are therefore removed.

```
array(['COMPLETE', 'START'], dtype=object)

array(['COMPLETE'], dtype=object)
```

6.2.2 FILTERING IRRELEVANT DATA: `W_VALIDEREN AANVRAAG`

To focus on specific segments of the process, I work with a DataFrame that exclusively includes cases concluding with the `W_Valideren aanvraag` activity. The idea here is to **remove anomalous or exceptional cases**, such as those that are excessively long or short, to concentrate on the **most representative cases**.




```
count    2744.000000
mean       40.813411
std        17.689472
min        19.000000
25%        28.000000
50%        36.000000
75%        49.000000
max       160.000000
dtype: float64
```

Method: Filtering W_Valideren aanvraag Cases

We calculated the **sequence lengths** (number of events per case). We then identified the **quartiles Q1 (25th percentile)**and **Q3 (75th percentile)** of these lengths.
Cases were segmented into three groups:

- **Short**: length ≤ Q1
- **Medium**: Q1 < length ≤ Q3
- **Long**: length > Q3

The primary idea is to mainly use the **"short" cases** because these represent situations where the process proceeded smoothly. The final activity `W_Valideren aanvraag` indicates a **positive outcome** (loan issued and the user is satisfied with the amount).
These short cases allow us to observe the **"ideal" process**, free from difficulties or delays. The **"medium" and "long" cases** will be retained for **targeted analyses** to identify bottlenecks, difficulties, or anomalous behaviors.

6.3 Process Discovery

For process discovery, I used inductive Petri nets with a threshold of 0.9.
I chose this approach because it:

- Guarantees a complete and deterministic model, free from deadlocks or invalid behaviors.
- **Allows for a good balance between accuracy and generalization**, thanks to the 0.9 threshold, which limits the inclusion of noise or overly rare cases.
- Produces models that are more readable and interpretable compared to other techniques.

I **did not use the other two main discovery techniques**, such as Alpha Miner and Heuristics Miner, because:

- **Alpha Miner** tends to produce incomplete or disconnected models, especially in the presence of noise or incomplete data, as seen in my dataset.
- **Heuristics Miner** generates more complex and less deterministic models, with many spurious transitions and unclear cycles, making interpretation and analysis difficult.

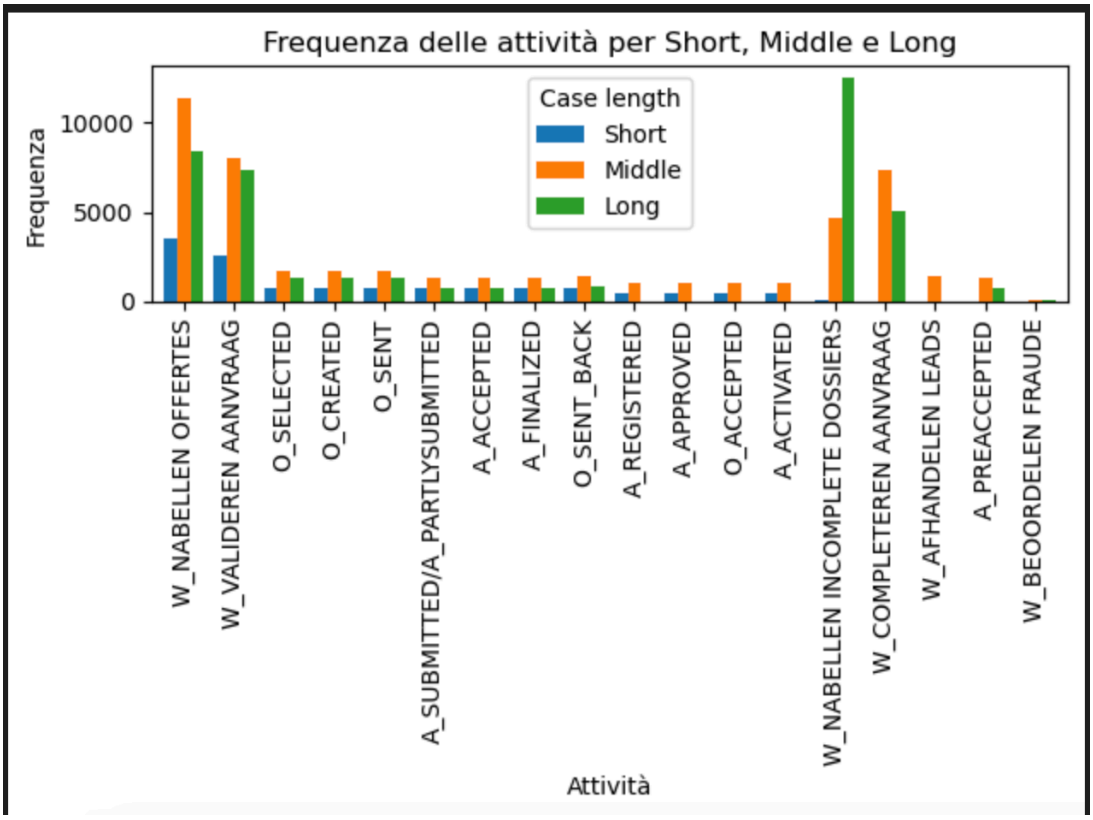
Initially, I applied the inductive method to the **complete dataset (df_W_complete)**, resulting in a very long Petri net characterized by numerous cycles.
Subsequently, I separately analyzed the **short cases**, which correspond to the smallest dataframes (25th percentile), and then the **middle and long cases**.
I observed that the process represented by the **short cases provides a clear idea of what the main process should look like**, even though it contains several cycles, which were further investigated later.
Analyzing the **middle and long cases**, I paradoxically found that the **long cases are much more similar to the short cases**, while the **middle cases exhibit many more activities and are generally longer**. All three versions show cycles predominantly in the same areas, corresponding to the **W_ transitions**.
Another interesting observation is that the W_boolean Fraud transition is absent in the short cases.

6.4 Descriptive analysis

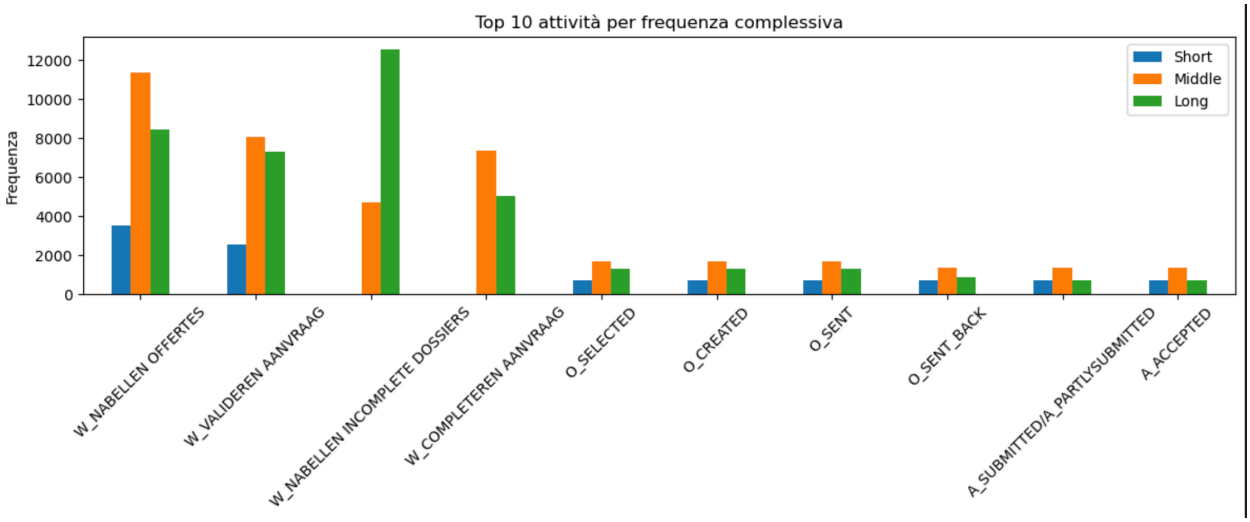
Descriptive analysis uses statistical tools to understand the distribution and characteristics of the data.

When analyzing the Petri net, numerous cycles emerged, and I aimed to understand which activities were causing them. For this reason, I examined the **frequency of activities** in each of the three available dataframes (**short, middle, and long**).

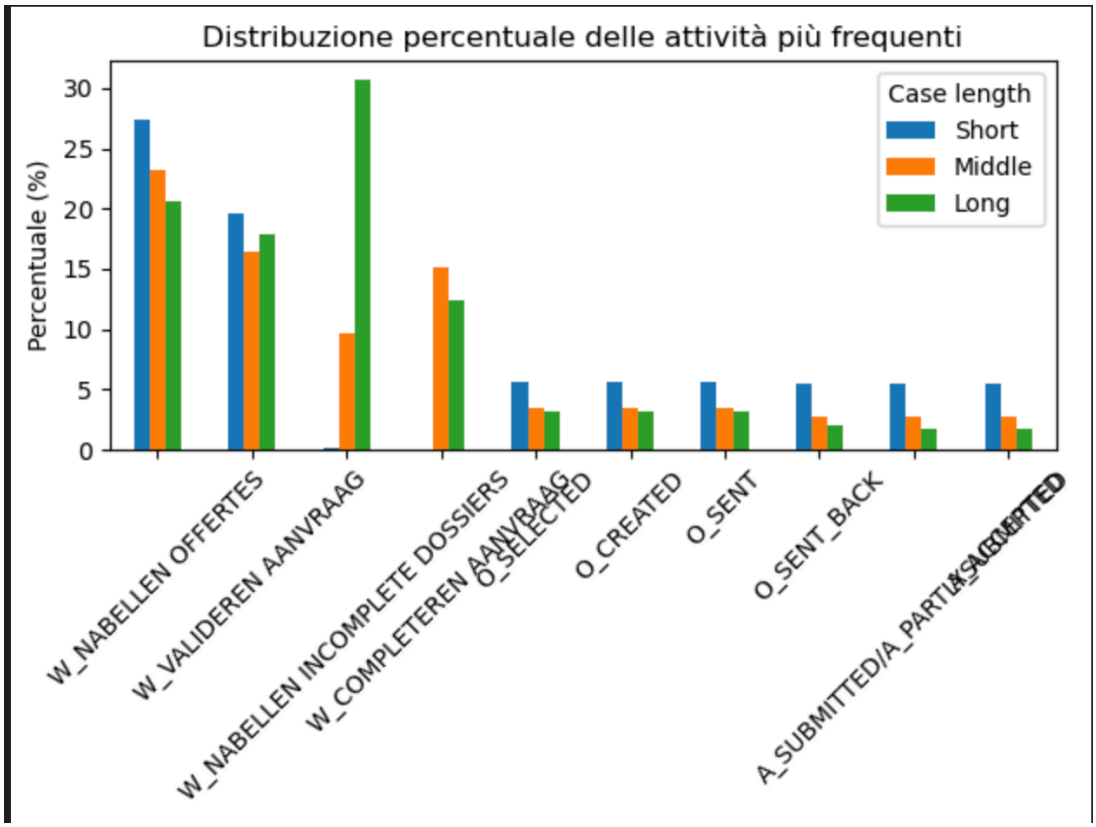
An interesting finding is that all three dataframes exhibit the same issue: a **high number of repeated activities**, which directly correspond to the cycles observed in the Petri nets.



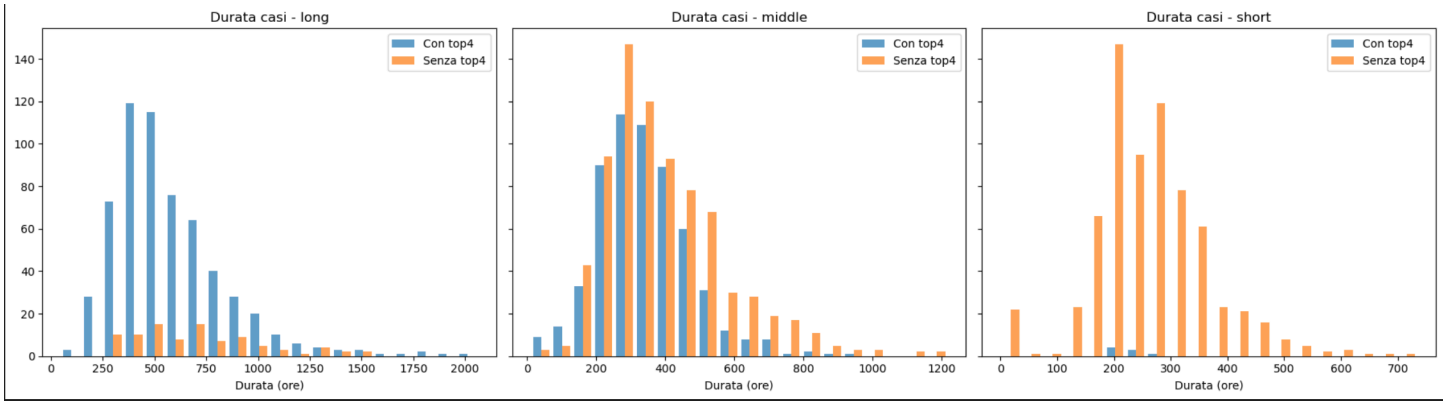
Next, I identified the **10 most frequent activities** for each dataframe and calculated their **percentage distribution**.



Analyzing the **percentage distribution of activities** is crucial for a normalized evaluation. Since the dataframes contain varying numbers of events, using absolute frequencies wouldn't allow for a direct and accurate comparison between different cases. Percentages, however, relate the frequency of each activity to the total number of events in the dataframe, providing a **relative measure**. This makes it easy to compare the importance and incidence of activities across processes of different sizes, and also facilitates the identification of common patterns or significant differences.



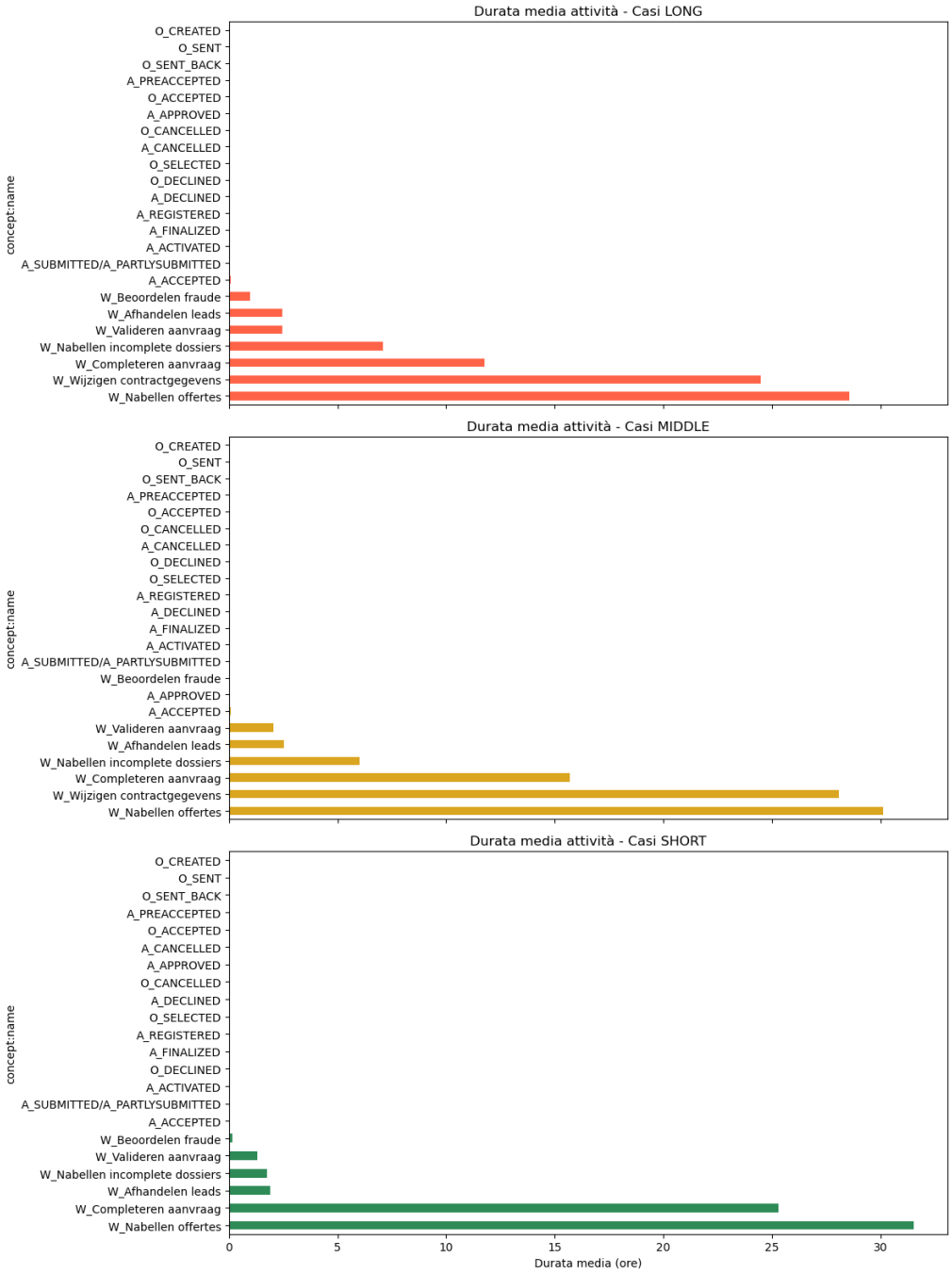
Among all activities, the four with the highest frequency all have the "W_" prefix. I therefore wanted to verify how the simultaneous presence of these four activities impacted the overall duration of cases, comparing them with cases where one or none of these activities were present.
The analysis shows that, in medium and short cases, it is not the quantity of these activities that increases the overall process duration.



To delve deeper, I calculated the **average duration for each activity**. Although a specific `average_activity_duration` function isn't explicitly defined in the provided code, this type of calculation is generally performed as follows:

1. For each case and each activity, the **start and end timestamps are identified**.
2. The difference between these timestamps is calculated, yielding the duration of each execution.
3. All durations are **grouped by activity**.
4. The **average of these durations is calculated**, providing the average duration for each activity.

In every instance, the time is higher for the four most frequent activities.



However, the initial calculation wasn't entirely accurate. So, I redefined the time metrics as follows:

- **Execution time**: The difference between the **`complete`** and **`start`** timestamps of the same activity within the same case. This represents the actual time spent executing the activity.
- **Waiting time**: The time elapsed between the end of the preceding activity and the beginning of the current one. This is calculated as the difference between the **`start`** timestamp of the current activity and the **`complete`**timestamp of the previous activity.
- **Total time**: The sum of waiting time and execution time, or the difference between the **`complete`** timestamp and a reference event (e.g., the start of the process or the previous activity).

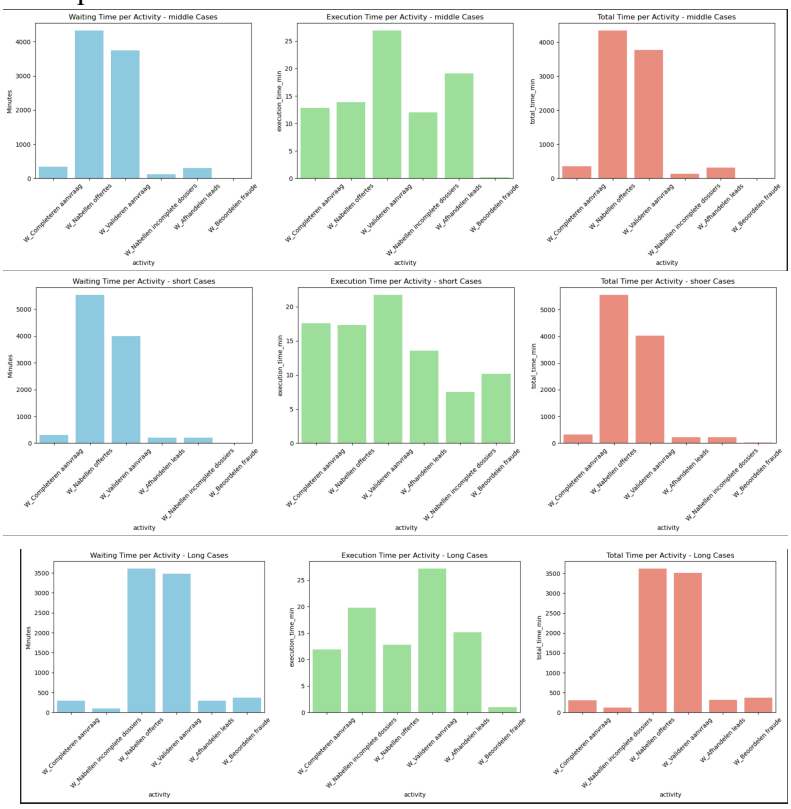
These calculations were applied to each dataframe (**long, middle, and short**), yielding more precise results that clearly confirm the bottlenecks represented by the following activities:

- **W_Completeren aanvraag**: Completion of information
- **W_Nabellen offertes**: Follow-up after sending offers
- **W_Valideren aanvraag**: Final validation of the request
- **W_Nabellen incomplete dossiers**: Retrieval of missing information
- W_Beoordelen fraude: Fraud control

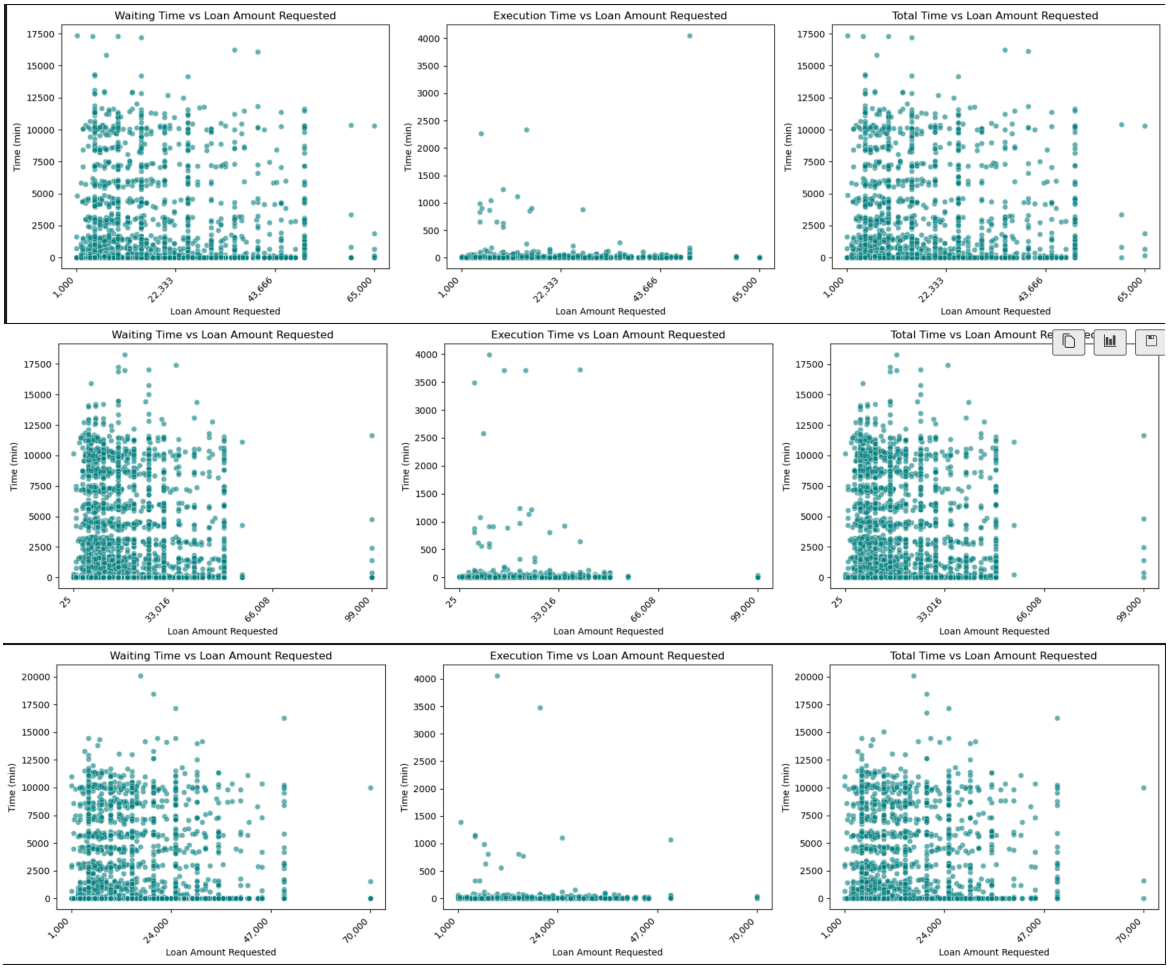
The **`W_Beoordelen fraude`** activity is practically absent or very rare, indicating that there were no requests for financial modifications or contractual variations. These steps are therefore primarily linked to obtaining the initial contract. This is also confirmed by the absence of the activity:

- **W_Wijzigen contractgegevens**: Modification of contract data

This activity never appears in the data. We can therefore assume with good certainty that there were no subsequent contractual modifications.



Another aspect I wanted to analyze was whether the duration of the process depended on the **requested loan amount**. In other words, I aimed to understand if higher amounts led to longer processing times. To investigate this, I broke down the analysis for each dataframe (**short, middle, and long**) and for each of the temporal metrics considered: **waiting time, execution time, and total time**.

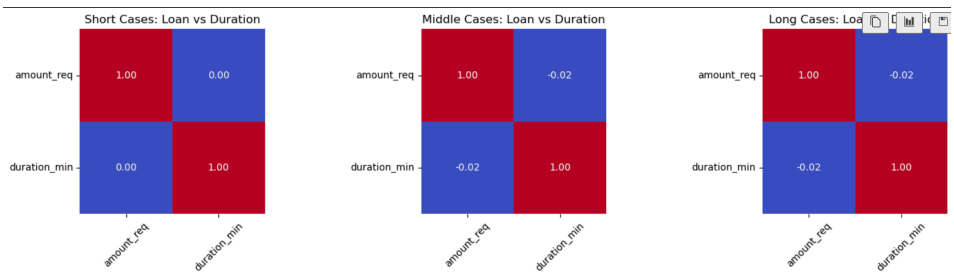


The results clearly show there's no significant correlation between the **requested amount** and the **process duration**. In essence, even for high-value loans, you can observe short processing times, and vice versa. This indicates that the financial size of the loan isn't a determining factor for the overall duration.

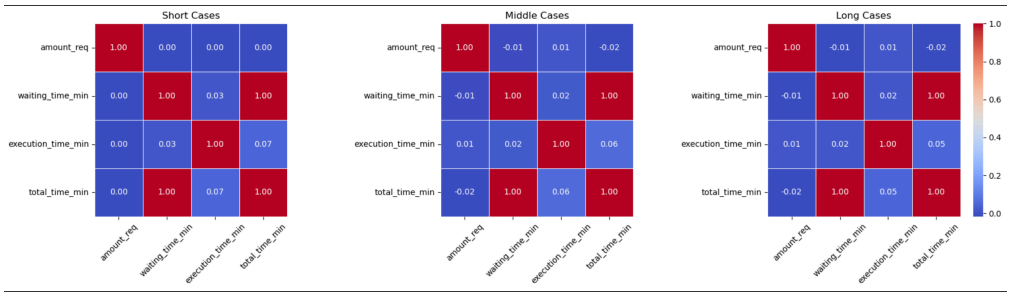
In detail:

- **Red** indicates a strong positive correlation (as one value increases, the other also increases).
- **Blue** indicates a negative correlation (as one value increases, the other decreases).
- **White** indicates no correlation.

The analysis, therefore, confirms an **absence of correlation (white area)** between the amount and the measured times.



However, the analysis reveals a **significant positive correlation between waiting time and activities with the "W_" prefix** (those we've already identified as potential bottlenecks). This confirms that the **waiting time between activities is a relevant factor** in determining the overall process duration, especially for those control, validation, and follow-up activities.
In summary, while the loan amount doesn't affect processing times, the waiting time between activities related to controls and verifications (the W_ activities) plays a crucial role in extending the total duration of cases.



In In conclusion, it's crucial to delve deeper into the specific issues related to activities with the "W_" prefix, as these significantly impact both waiting times and the overall duration of cases. Understanding the root causes of these delays—such as inefficiencies, lack of resources, redundant steps, or the need for additional verifications—is essential for targeted interventions.
A more detailed analysis could help identify the true bottlenecks and develop strategies to streamline the process, thereby reducing both the length and complexity of cases and improving overall workflow efficiency.

6.6 Conformance checking

To identify deviations from the typical sequence of events, we used **conformance checking**.
We employed the **token-based replay method** from the **PM4Py library** to analyze three groups of cases against the process model. The calculated metrics are:

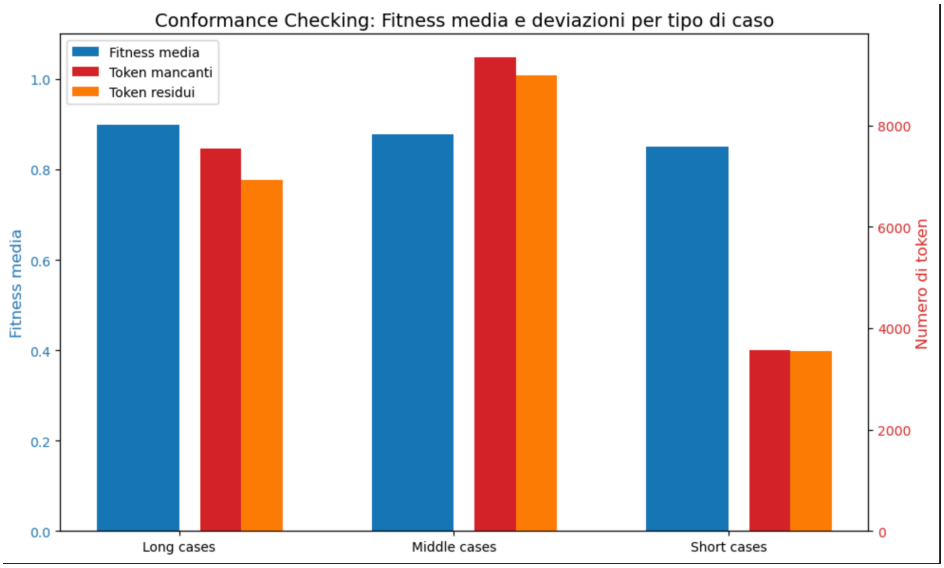
- **Average fitness**: Measures how well the traces fit the model (ranging from 0 to 1).
- **Missing tokens**: Elements expected by the model but absent in the traces, indicating deviations.
- **Remaining tokens**: Elements present in the traces but not consumed by the model, also signals of deviation.
- Total number of traces analyzed for each group.

Tipo di caso	Fitness media	Token mancanti totali	Token consumati totali	Token residui totali	Numero tracce
Long	0.90	7.546	71.153	6.932	689
Middle	0.88	9.340	71.867	8.985	1.353
Short	0.85	3.564	22.579	3.549	702

Visualization and Analysis of Results
To facilitate comparison, we created a **bar chart with two axes**:

- **Left axis (blue)**: Average fitness per group.
- **Right axis (red/orange)**: Missing tokens and remaining tokens per group.

This allows us to observe how **conformity varies among the groups**, correlating adherence to the model with deviations.
The importance of this analysis lies in its ability to pinpoint which traces deviate from the model, helping us understand which groups exhibit greater problems and identify potential causes of inefficiency.
Key results
The **average fitness decreases from long to short cases**, indicating that shorter cases deviate more from the model. **Missing and remaining tokens are higher in the middle cases**, suggesting significant deviations even in those cases.



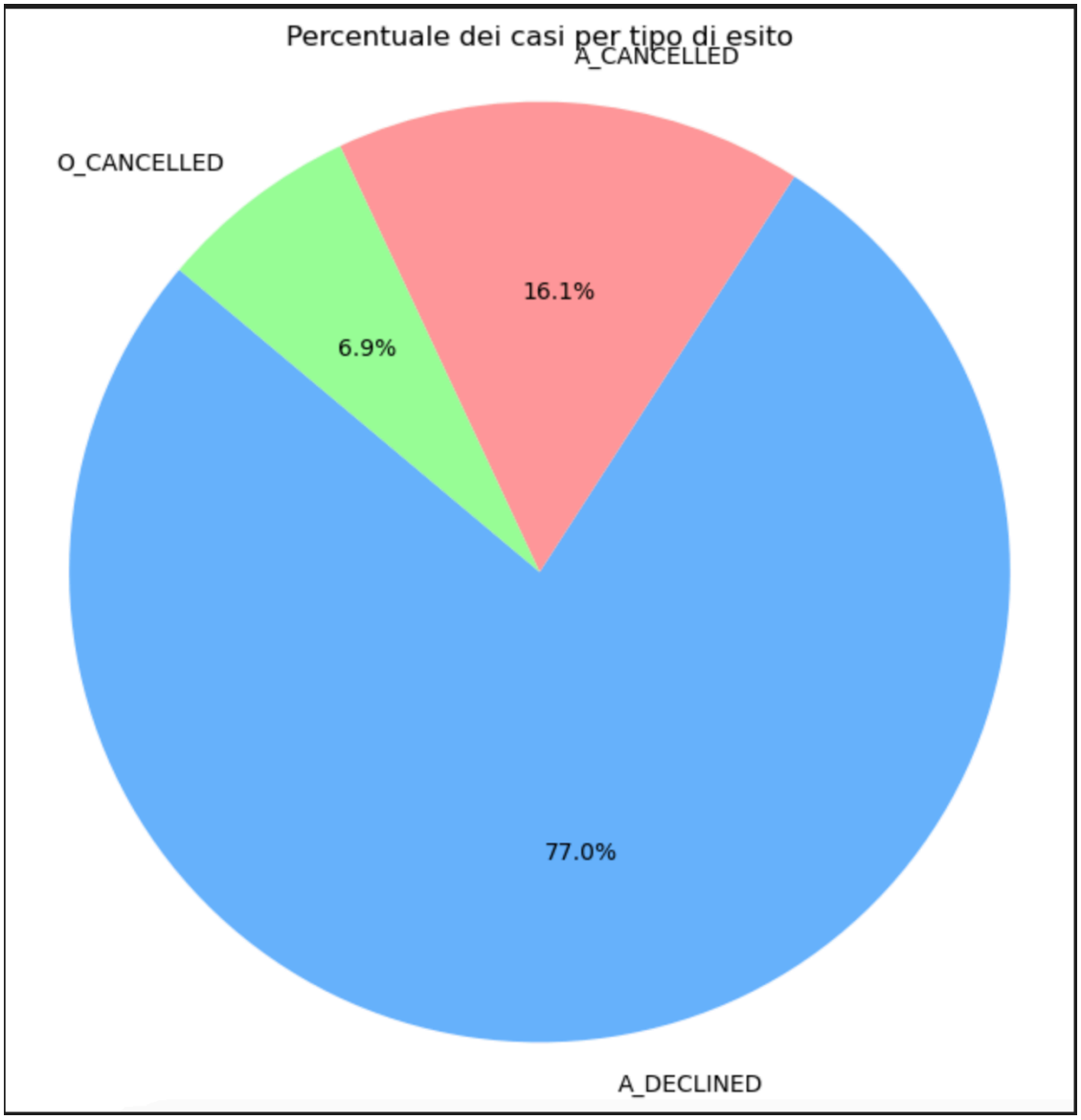
6.2.2 FILERING IRRELEVANT DATA DF_AO_DC

Here, I've segmented the dataset based on the final activities related to **rejected cases**, which include the events **A_DECLINED, A_CANCELLED, and O_CANCELLED** (the latter, although not among the primary expected outcomes, is relevant for future analysis).

The **A_APPROVED** event never appears as a final event. For this reason, it makes no sense to artificially force a case closure based on this event, to avoid considering still open or incomplete activities as closed. As anticipated, I observed that the **A_DECLINED** activity occurs most frequently as a final event.

Therefore, I further divided the dataframe into three parts using filters on the final activities:

df_declined
df_cancelled
df_o_cancelled



6.3 Process Discovery

I opted for **inductive Petri nets with a 0.9 threshold** for process discovery. This choice was deliberate because it:

- Ensures a complete and deterministic model, avoiding deadlocks or invalid behaviors.
- **Achieves a good balance between accuracy and generalization**, as the 0.9 threshold helps to exclude noise or very rare cases.
- Generates models that are more readable and easier to interpret compared to other methods.

I specifically avoided **Alpha Miner** and **Heuristics Miner** because:

- **Alpha Miner** often produces incomplete or disconnected models, especially with noisy or incomplete datasets like yours.
- **Heuristics Miner** tends to create more complex and less deterministic models, with many spurious transitions and unclear cycles, making them harder to interpret and analyze.

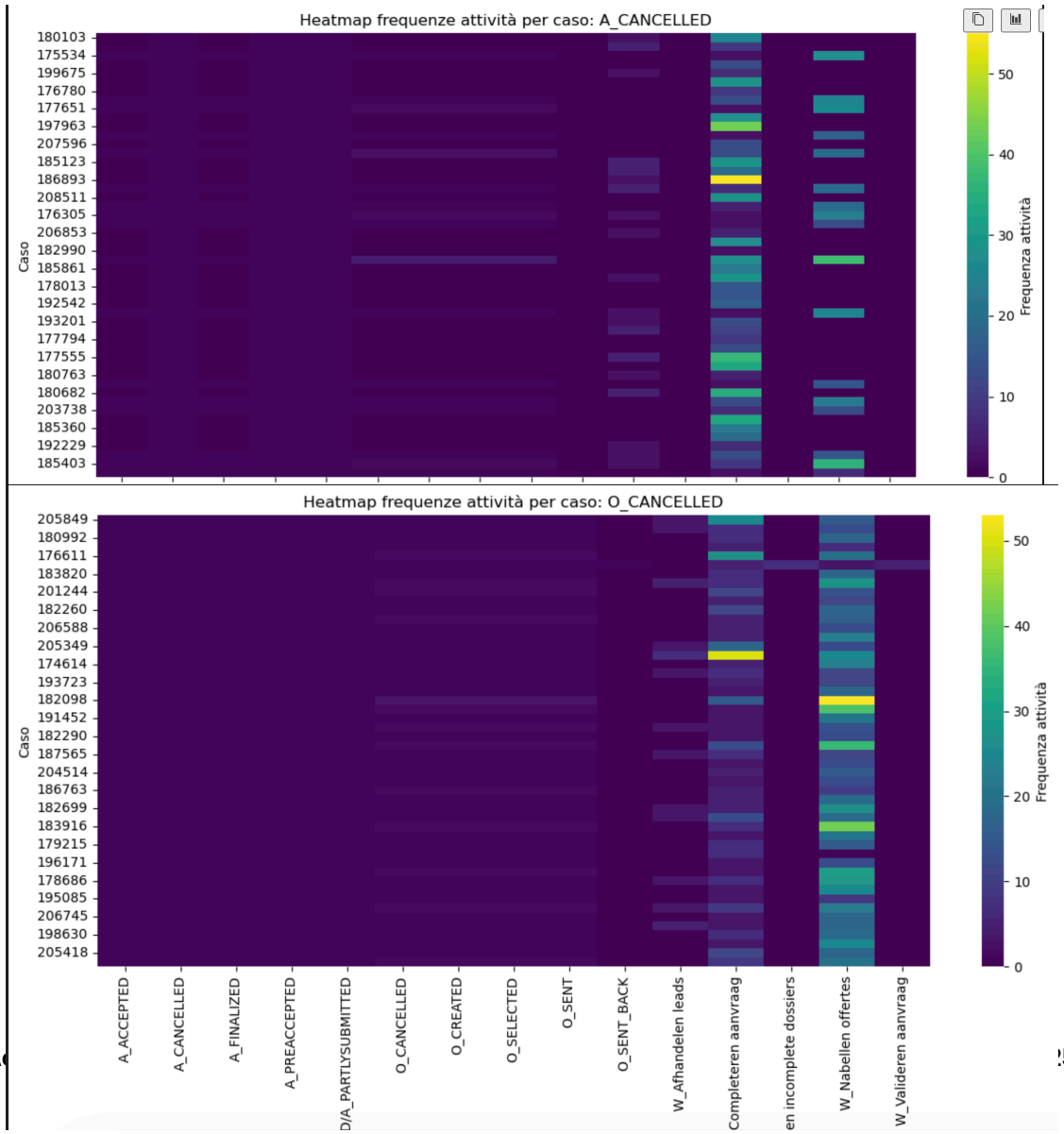
I applied the inductive Petri net method to three subsets of your dataset. For the **A_DECLINED** activity, you obtained a very simple model with just two activities: the common initial activity (**A_SUBMITTED**) and the final activity (**A_DECLINED**).
I didn't use noise filtering thresholds in this instance, suggesting that this model likely represents application errors where an activity is created and then immediately removed. These events can be considered **unnecessary noise and were subsequently removed** from your dataset; I didn't use that subset further.
Instead, you focused on a more detailed analysis of cases with **O_CANCELLED** final activities (cancellation or rejection of the offer).
The second category, cases ending with **A_CANCELLED**, presented a much more elaborate model. This model included various offer-related activities and waiting activities (**W_**), which can help to understand why the request might have been rejected by the application or cancelled (similar to **O_DECLINED** and **A_DECLINED**).

6.4 Descriptive analysis

To pinpoint activities that might indicate a high probability of rejection, I created a **heatmap** analyzing the average frequency of activities in cases that end with a negative outcome.

Key Findings

- In cases concluding with **O_CANCELLED** (offer cancelled), the activity with the highest value is **W_Nabellen offertes**, which represents the follow-up after sending offers.
  The term "follow-up" here implies a re-contact or reminder to the client, for instance, to prompt a decision after an offer has been sent. This suggests that when multiple follow-ups are necessary, there's a **higher likelihood that the client will abandon the process or cancel the offer.**
- In cases ending with A_DECLINED (application declined), the most recurring activity is W_Completeren aanvraag, which is the completion of necessary information for the application.
  This may indicate that in cases where multiple interventions are required to complete the application, there's a greater probability that the request will ultimately be rejected.



The two box plots compare the number of activities performed before cancellation in **A_CANCELLED** and **O_CANCELLED** cases.

A_CANCELLED

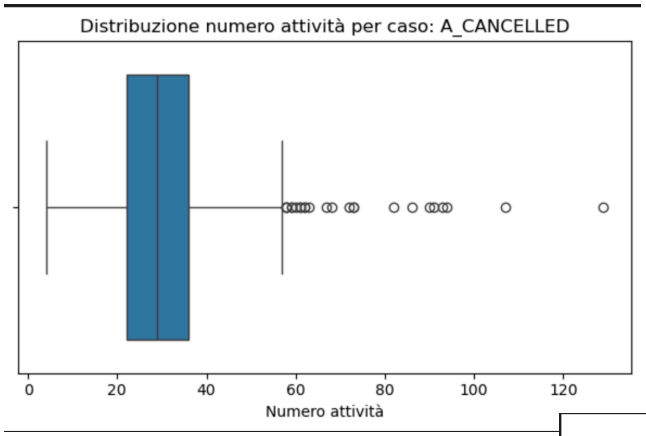
- **Median**: 35 activities. This means half of these cases are cancelled relatively early in the process.
- **Interquartile Range (IQR)**: Approximately 25–35 activities. The majority of cases have a limited number of activities before cancellation.
- **Outliers**: Numerous cases with over 60 activities, some exceeding 120. These indicate exceptions that are significantly longer.
- **Minimum**: Cancellation can occur after very few activities, suggesting some cases stop almost immediately.

O_CANCELLED

- **Median**: Approximately 42 activities. Cancellations generally happen later in these cases.
- **IQR**: Approximately 35–45 activities. These processes are slightly longer compared to A_CANCELLED cases.
- **Outliers**: Also present here, but they typically stop around 90 activities.
- **Minimum**: The minimum number of activities is higher than in A_CANCELLED cases, meaning fewer "early" cancellations.

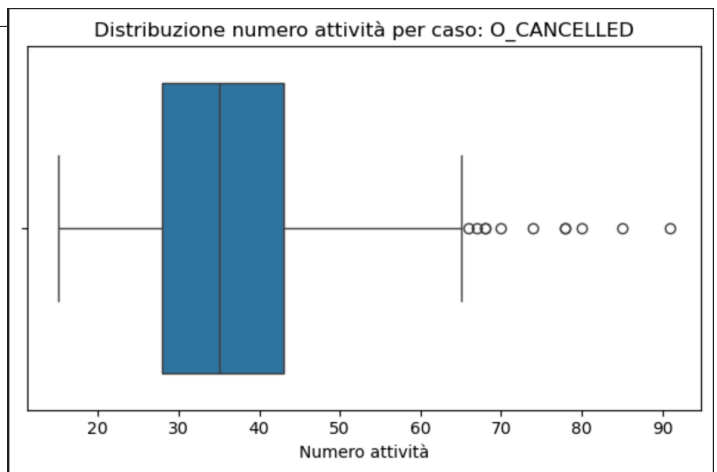
Comparison

- **O_CANCELLED** cases generally last longer on average compared to **A_CANCELLED** cases.
- Both types show outliers, but **A_CANCELLED** exhibits greater variability with more extreme tails.
- **A_CANCELLED** cancellations can occur very early or very late, while **O_CANCELLED** is more stable but still subject to exceptions.




```
--- Statistiche Boxplot per A_CANCELLED ---
Minimo (baffo inferiore): 4
1° Quartile (Q1): 22.0
Mediana (Q2): 29.0
3° Quartile (Q3): 36.0
Massimo (baffo superiore): 57
IQR: 14.0
Numero di outlier: 24
```

```
--- Statistiche Boxplot per O_CANCELLED ---
Minimo (baffo inferiore): 15
1° Quartile (Q1): 28.0
Mediana (Q2): 35.0
3° Quartile (Q3): 43.0
Massimo (baffo superiore): 65
IQR: 15.0
Numero di outlier: 11
```



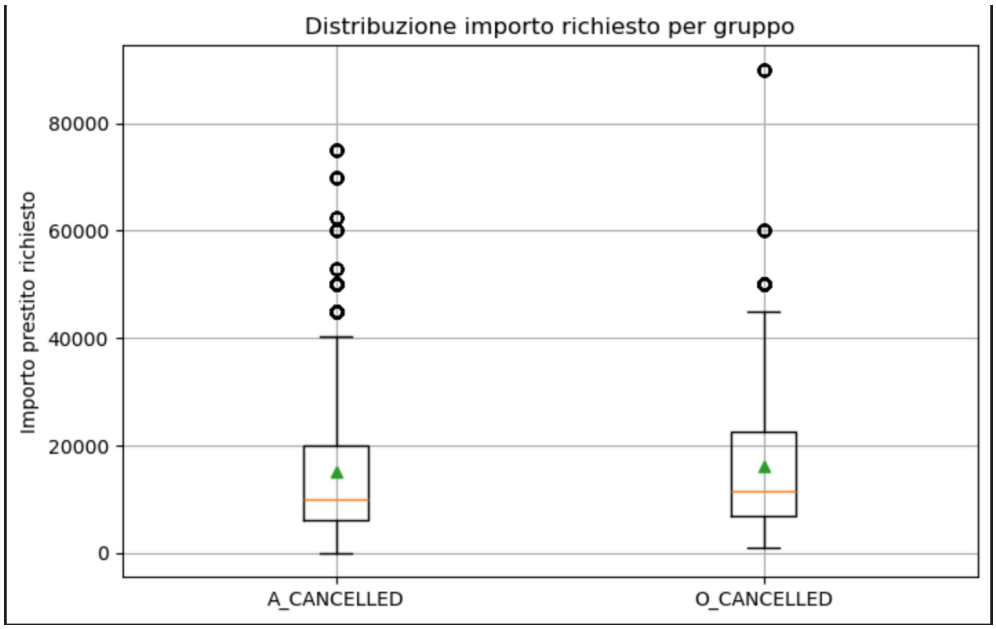
Statistical Analysis and Visualization of Loan Amount Distribution for A_CANCELLED and O_CANCELLED Groups

```
Statistiche per A_CANCELLED:
  min: 0
  max: 75000
  median: 10000.0
  mean: 15209.485712966809
  std: 12445.339828718335
  count_valid: 21663
Statistiche per O_CANCELLED:
  min: 1000
  max: 90000
  median: 11500.0
  mean: 16233.375254267268
  std: 12920.99438570207
  count_valid: 11307
```

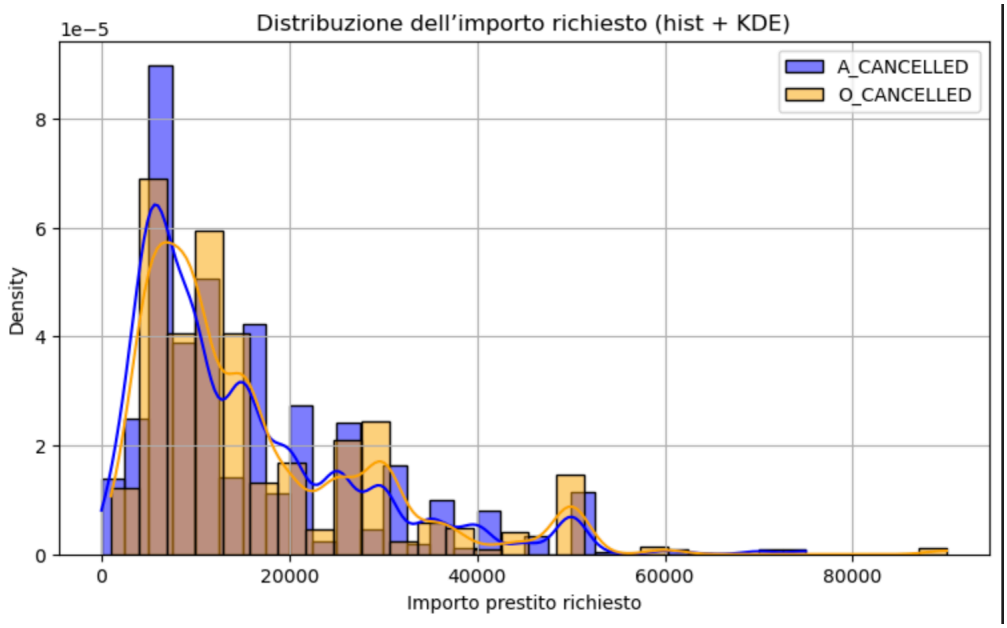
We've calculated the main descriptive statistics for the requested loan amounts in the two groups:

For the **A_CANCELLED** group, the minimum amount is **€0**, the maximum is **€75,000**, with a **median of €10,000** and an **average of approximately €15,200**. The high standard deviation (~€12,445) indicates good data variability. **21,663 cases** were considered valid.

For the **O_CANCELLED** group, the minimum amount is **€1,000**, the maximum is **€90,000**, with a **median of €11,500**and a slightly higher average of approximately **€16,200**. The standard deviation is similar to the first group (~€12,920), with **11,307 valid cases**.

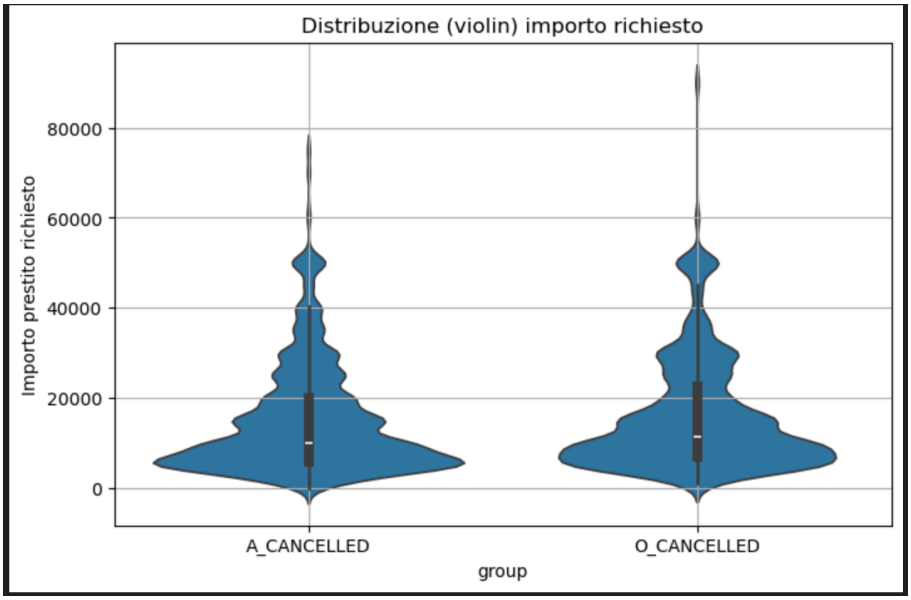


We explored the distributions using **histograms accompanied by Kernel Density Estimates (KDEs)**, which continuously show the shape and density of the data. Both groups exhibit a **peak in the amount range between €5,000 and €15,000**, with a long tail towards higher values. However, the **O_CANCELLED group tends to show a higher concentration on larger amounts** compared to A_CANCELLED.



Analysis with **violin plots** confirms these observations, highlighting the data distribution and density more intuitively. The width of the "violin" at a given point indicates the relative density of cases with that amount. Both groups show a similar distribution and consistent variability, with **O_CANCELLED presenting a slight prevalence of higher amounts and fewer cases near zero**. The quartiles are clearly highlighted and indicate how most of the data concentrates around the medians.

In summary, while showing similar characteristics, loans in the **O_CANCELLED group tend to have higher average amounts** compared to A_CANCELLED, with a distribution more shifted towards higher values and a lower presence of very low amounts. Both distributions, however, show significant variability with some extremely high amount cases.

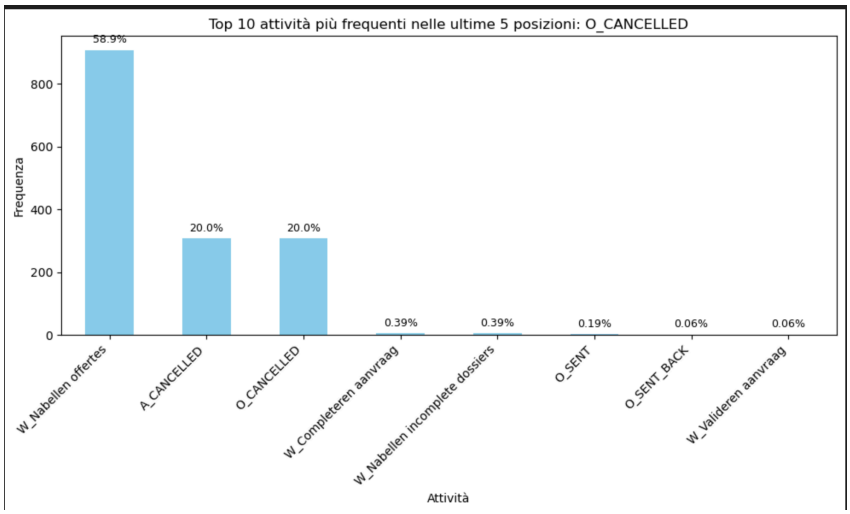


Analysis of Patterns in Last Activities Before Cancellation

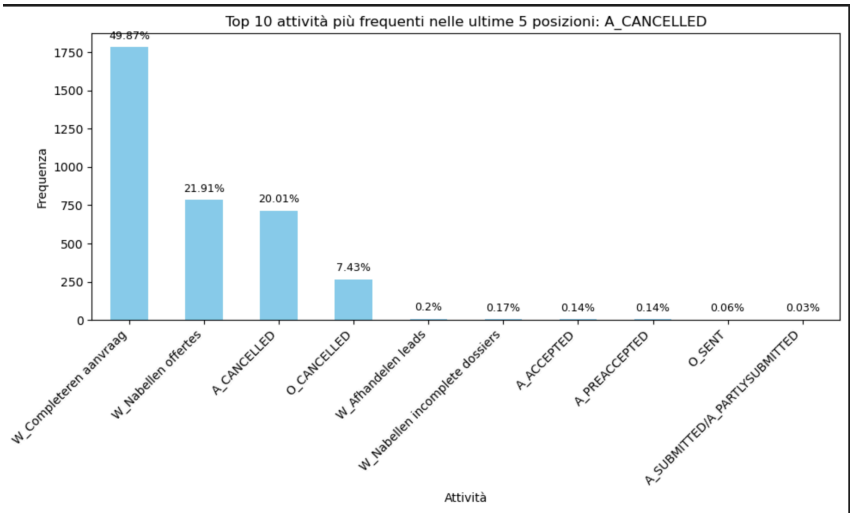
The goal of this analysis was to determine if observing the **last 5 activities of each case** could reveal recurring patterns that highly predict a case ending with a cancellation activity, such as **A_CANCELLED** or **O_CANCELLED**.

Specifically, we aimed to understand if the repeated presence of an **A_CANCELLED** or **O_CANCELLED** activity in the last positions could be predictive of the final cancellation. The results show a clear pattern: if one of these activities appears among the last ones, it is very likely that the case will conclude with that same cancellation activity.

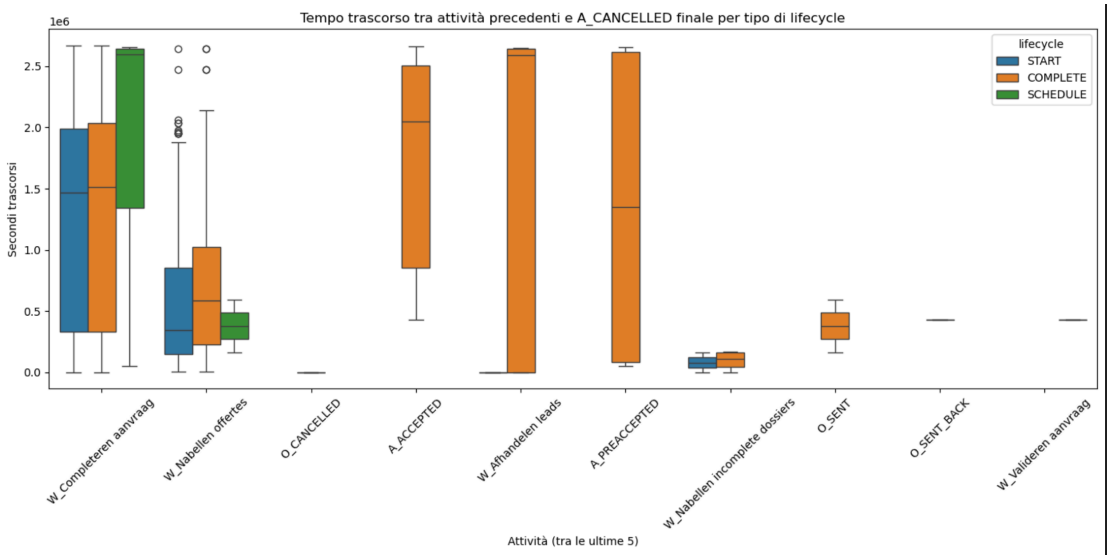
A particularly interesting finding concerns **O_CANCELLED**: when **W_Nabellen offertes** (follow-up after sending offers) appears among the last activities, its frequency is very high. This suggests a strong and repetitive pattern. Indeed, for approximately **60% of cases that conclude with O_CANCELLED**, this specific activity appears among the last ones, indicating a clear trend.



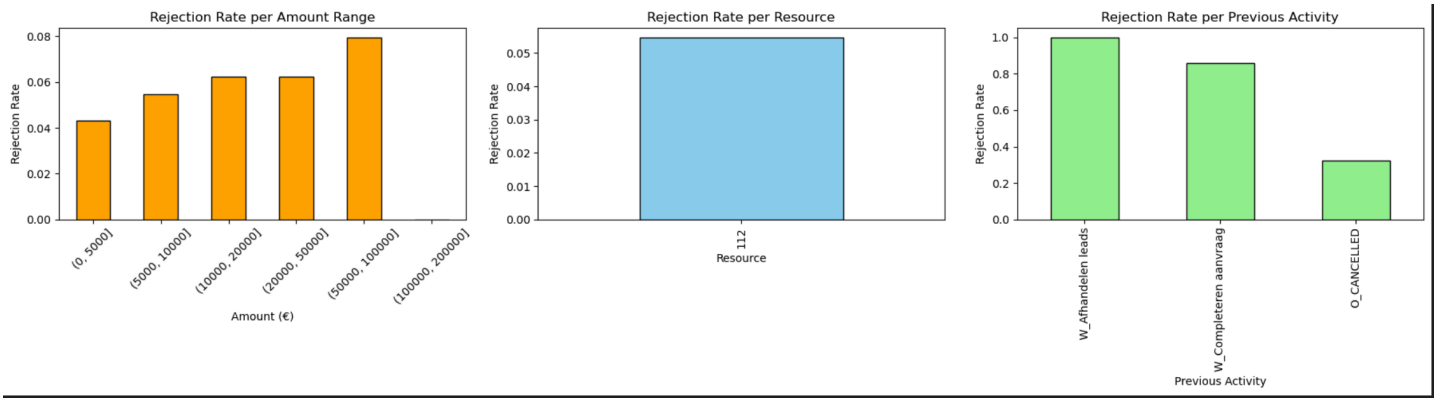
In the case of **A_CANCELLED**, the most frequent activity among the last ones is **W_Completeren aanvraag: completion of information**. While this might be less intuitive, it's equally significant: nearly **50% of cases ending with A_CANCELLED show this activity among their final steps**, suggesting that even in this scenario, there's a typical sequence leading to cancellation.

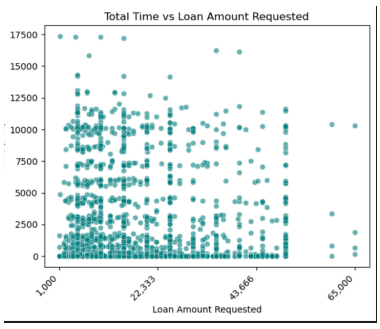


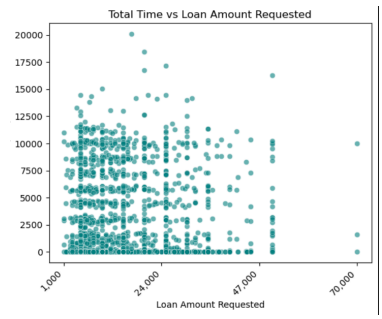
Analysis of Time and Transition Type (Lifecycle)
To delve even deeper, we also analyzed the time elapsed between the last activities and the final cancellation, distinguishing by **transition types (schedule, start, complete)**.



Finally, an analysis of **rejection rates** was conducted, revealing that cases belonging to the **W_ category** exhibit a **high rejection rate**. This confirms previous observations: W_ cases are not "neutral" or passive containers; instead, they are often subjects of rejection. Specifically, the element with the **highest rejection rate is '112'**, which corresponds to a department or an automated activity. It's also noted that **high-amount loans are frequently rejected**.
Conversely, in other datasets, the distribution of rejections is more concentrated on central values, ranging between €1,000 and €50,000, meaning not excessively high loan amounts.







6.6 Conformance checking

In this case, I didn't perform a **conformance checking analysis**, as I believe there isn't a real "win-win" benefit in applying it.

The data considered represents cases where the application was **rejected** (either by the system or the user), so it doesn't describe an "ideal" process flow to be compared against a reference model. Conformance checking generally serves to identify deviations from the typical sequence of events. However, in this context, the entire process can be considered a deviation in itself, as it ends in an undesirable outcome for both parties:

- **For the bank**: There's a loss of a potential customer, and thus a missed opportunity for profit and customer loyalty. Additionally, the positive effect of word-of-mouth marketing, which remains one of the strongest marketing channels due to trust among acquaintances, is lost.
- **For the customer**: The request is rejected, and if they still desire the loan, they'll have to repeat the entire process from scratch, incurring additional time and effort.

In summary, these cases aren't well-suited for conformance analysis because there's no "expected process" in which such outcomes are considered successes or objectives to be replicated.

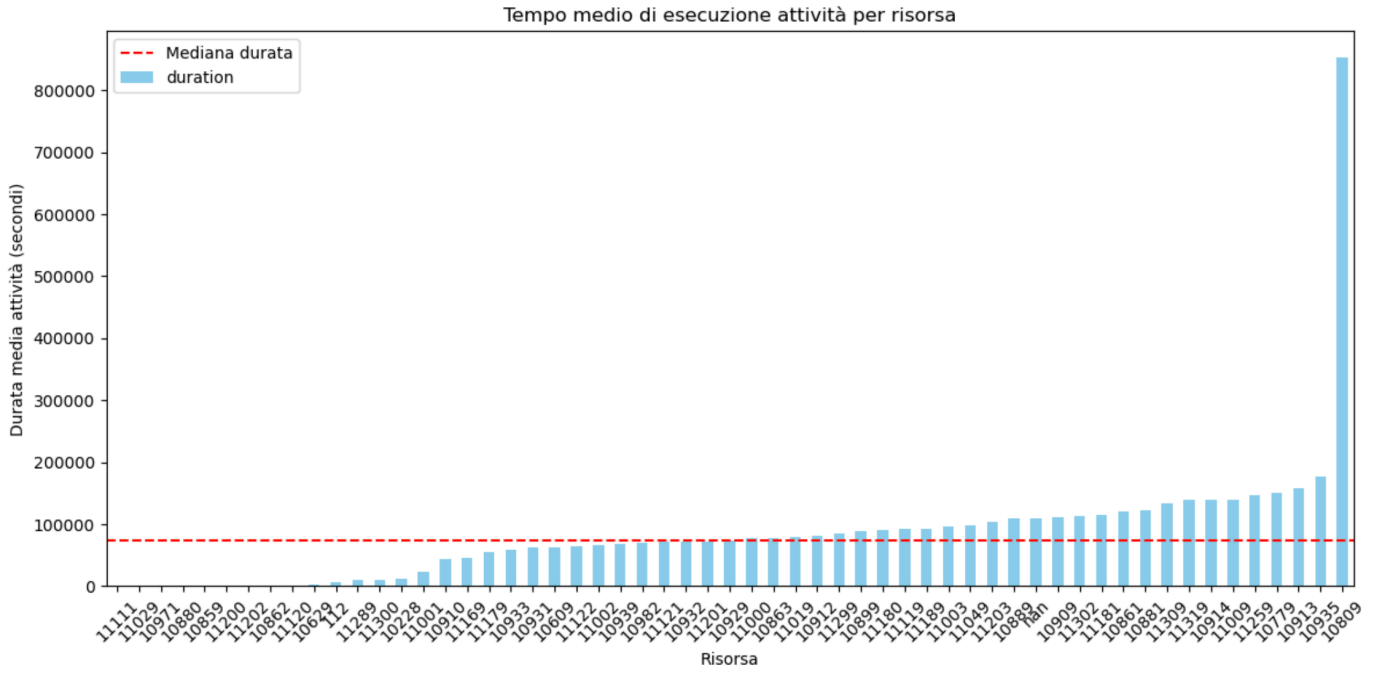
6.7 Intervention strategies

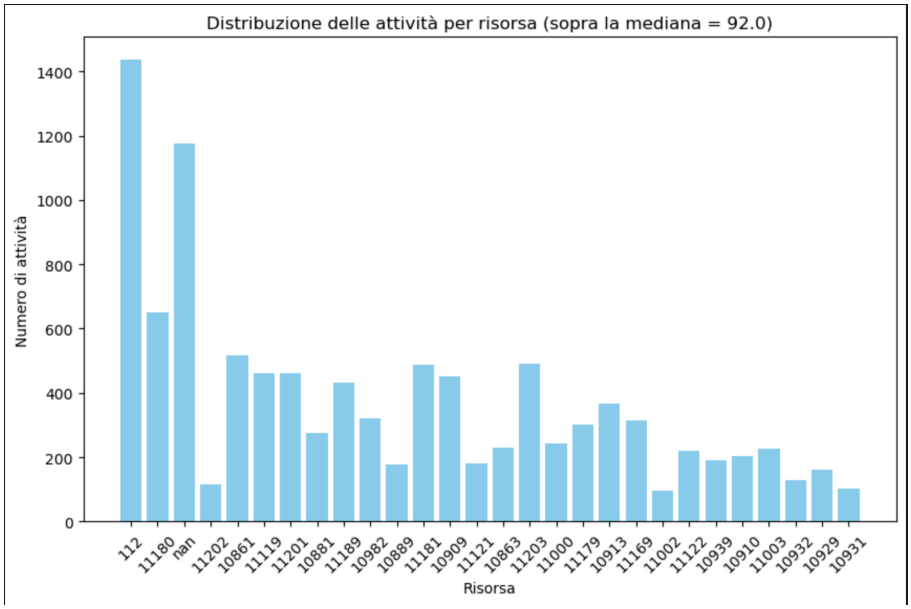
Itervention strategies are developed based on the information and knowledge gained from previous analytical steps.
These strategies aren't static; they're continuously refined through ongoing monitoring and the measurement of results. In practice, it's an iterative cycle where:

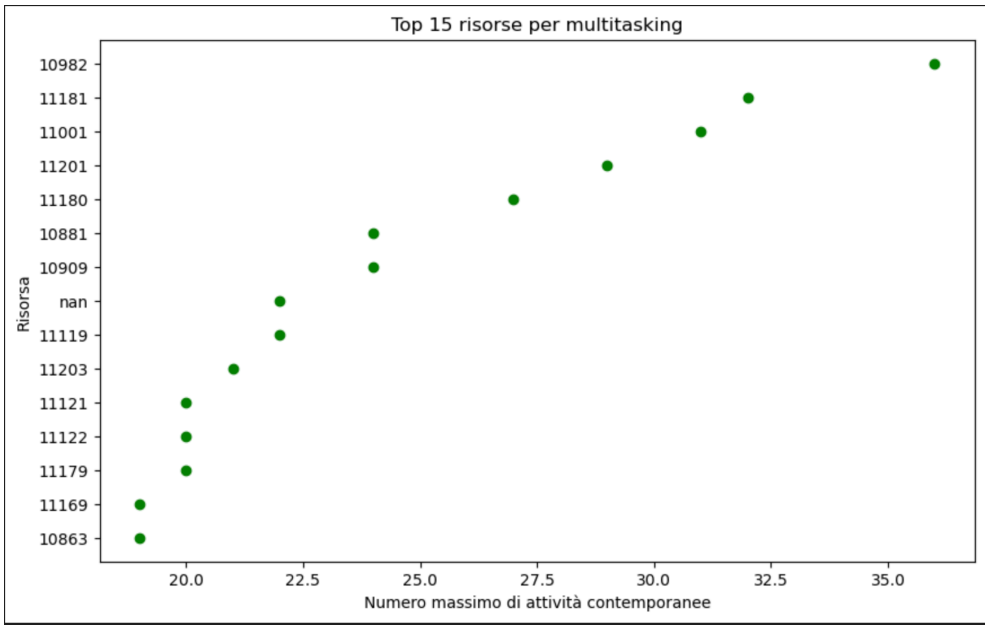
- **Collected data and analyses** guide the definition of interventions to be implemented (e.g., structural changes to the process, reorganization of activities, staff training).
- Once implemented, the **strategies are monitored** to assess their effectiveness over time.
- Based on the **results obtained and feedback received**, further adjustments are made to optimize processes and address any remaining critical issues.

This approach ensures that actions taken are always aligned with continuous improvement goals, also allowing for a timely response to new challenges or changes in the operational context.
Finally, to support the development of these strategies, I calculated:

- The **average execution time per resource**, to identify potential bottlenecks or operational overloads.
- The **distribution of activities per resource**, focusing on those exceeding the median, to highlight any anomalous load concentrations.
- The **top 15 multitasking resources**, to evaluate potential effects on execution time and the quality of work performed.







6.7.1 FUTURE DEVELOPMENTS

One potential future development involves **predictive analytics**, aiming to **identify cases at risk of rejection early in the process**. This would allow for **targeted interventions** (like personalized reminders or dedicated assistance) to increase the likelihood of application success.

Furthermore, we could deepen the **analysis of activity sequences using pattern recognition techniques**. This would help to pinpoint behavioral configurations that consistently lead to negative outcomes, thereby enhancing **decision support**.

https://github.com/IkramBourras/Business-information-System