Process Mining Project

by Leo Mastronardi

A.Y. 2025/2026

Index

1 Introduction

Process mining is a data-driven discipline that analyzes business processes using the event data recorded by information systems. Instead of relying only on predefined or "idealized" workflow models, process mining reconstructs how processes are actually executed. By extracting knowledge from event logs, it becomes possible to visualize real behavior, evaluate process quality, detect deviations, and identify improvement opportunities.
An *event log* is the fundamental input for process mining. It is composed of cases (process instances), each represented by a sequence of events ordered in time. Every event typically contains at least a case identifier, an activity name, and a timestamp. From this structured data, algorithms can discover a process model that describes the control-flow relations between activities.

Process discovery techniques generate models directly from the event log without prior assumptions. These models provide a compact representation of observed behavior. In this project, discovered processes are represented as Petri nets, a formal modeling notation widely used in process mining.
A *Petri net* describes the dynamics of a process using places (states/conditions), transitions (activities), and tokens (flow of control). This formalism allows both visualization and quantitative evaluation through conformance and quality metrics.

This work applies a simple process mining pipeline to the Road Traffic Fine Management Process dataset. The analysis includes log exploration, optional filtering to focus on dominant variants, process discovery using three algorithms (Alpha Miner, Heuristics Miner, Inductive Miner), and model evaluation based on fitness, precision, generalization, and simplicity. Additionally, a dashboard was developed to visualize results and integrate an LLM assistant for automated reporting, interactive questions, and basic anomaly interpretation. The objective of the project is to implement and document a clear, interpretable workflow that demonstrates how process mining methods can be applied to a real event log, from data preparation to model analysis and decision support.

2 Dataset

The dataset selected for this project is the Road Traffic Fine Management Process event log, provided in XES format. (Road Traffic Fine Management Process (dataset))
The log records the execution of activities related to the lifecycle of road traffic fine management cases. Each case represents a specific fine, while events capture the activities performed over time, including their timestamps. The dataset follows the standard structure required for process mining, containing at least a case identifier (case:concept:name), an activity name (concept:name), and a timestamp (time:timestamp).

2.1 Preprocessing and Filtering

This section describes the operations performed to load, inspect, and optionally simplify the event log before applying discovery and evaluation techniques.

The event log is loaded directly from the XES file using the PM4Py library. The log is then converted into a pandas DataFrame to facilitate statistical inspection and filtering operations. Output directories are created to store visualizations and evaluation results for both the raw and filtered logs.

2.1.1 Data Overview

An initial exploratory analysis is performed on the raw log. The following statistics are computed:

- Number of cases
- Number of events
- Number of distinct activities
- Time range (minimum and maximum timestamp)
- Most frequent activities
- Variant distribution

Variants are obtained by grouping events by case identifier and representing each case as an ordered sequence of activities. This overview provides a basic understanding of process complexity, frequency patterns, and behavioral variability. The same analysis is repeated after filtering to observe how the simplification step affects the log characteristics.

```
=== RAW LOG OVERVIEW ===
cases: 150370 | events: 561470 | activities: 11
time range: 1999-12-31 23:00:00+00:00 -> 2013-06-17 22:00:00+00:00

Top activities:
concept:name
Create Fine                               150370
Send Fine                                 103987
Insert Fine Notification                   79860
Add penalty                                79860
Payment                                    77601
Send for Credit Collection                 59013
Insert Date Appeal to Prefecture            4188
Send Appeal to Prefecture                   4141
Receive Result Appeal from Prefecture        999
Notify Result Appeal to Offender             896

variants: 231
top variants (first 5):
- Create Fine -> Send Fine -> Insert Fine Notification -> Add penalty -> Send for Credit Collection | cases: 56482
- Create Fine -> Payment | cases: 46371
- Create Fine -> Send Fine | cases: 20385
- Create Fine -> Send Fine -> Insert Fine Notification -> Add penalty -> Payment | cases: 9520
- Create Fine -> Send Fine -> Insert Fine Notification -> Add penalty -> Payment -> Payment | cases: 3736

avg events per case: 3.73
```

Overview of the raw log

```
=== FILTERED LOG OVERVIEW ===
cases: 136494 | events: 485938 | activities: 6
time range: 1999-12-31 23:00:00+00:00 -> 2013-06-17 22:00:00+00:00

Top activities:
concept:name
Create Fine                   136494
Send Fine                      90123
Insert Fine Notification       69738
Add penalty                    69738
Payment                        63363
Send for Credit Collection     56482

variants: 5
top variants (first 5):
- Create Fine -> Send Fine -> Insert Fine Notification -> Add penalty -> Send for Credit Collection | cases: 56482
- Create Fine -> Payment | cases: 46371
- Create Fine -> Send Fine | cases: 20385
- Create Fine -> Send Fine -> Insert Fine Notification -> Add penalty -> Payment | cases: 9520
- Create Fine -> Send Fine -> Insert Fine Notification -> Add penalty -> Payment -> Payment | cases: 3736

avg events per case: 3.56
Cases kept: 136494 / 150370 (90.77%)
Events kept: 485938 / 561470 (86.55%)

Removed activities:
- Appeal to Judge
- Insert Date Appeal to Prefecture
- Notify Result Appeal to Offender
- Receive Result Appeal from Prefecture
- Send Appeal to Prefecture
```

Overview of the filtered log

2.1.2 Variant-Based Filtering

To focus the analysis on dominant process behavior, a variant-based filtering has been applied. Cases are grouped into variants based on their activity sequences. Variants are sorted by frequency, and only the most frequent variants are retained until they cumulatively cover 90% of all cases. All cases belonging to the selected variants are kept, while cases associated with infrequent variants are removed. This produces a filtered log that emphasizes common execution paths and reduces noise and structural complexity.
This filtering step is intended to obtain a clearer view of typical process behavior rather than to improve model performance artificially. Consequently, results derived from the filtered log are interpreted as representing “common scenarios” rather than the full variability of the original process.

3 Process Discovery

Process discovery aims to automatically derive a process model from an event log without using prior knowledge about the underlying workflow. The discovered model represents the control-flow relations observed in the recorded executions. In this project, discovery is performed on both the raw log and the filtered log (common scenarios), enabling the analysis of full and dominant behavior perspectives. In addition to Petri net models, a Directly-Follows Graph (DFG) is generated as a preliminary representation of process behavior.

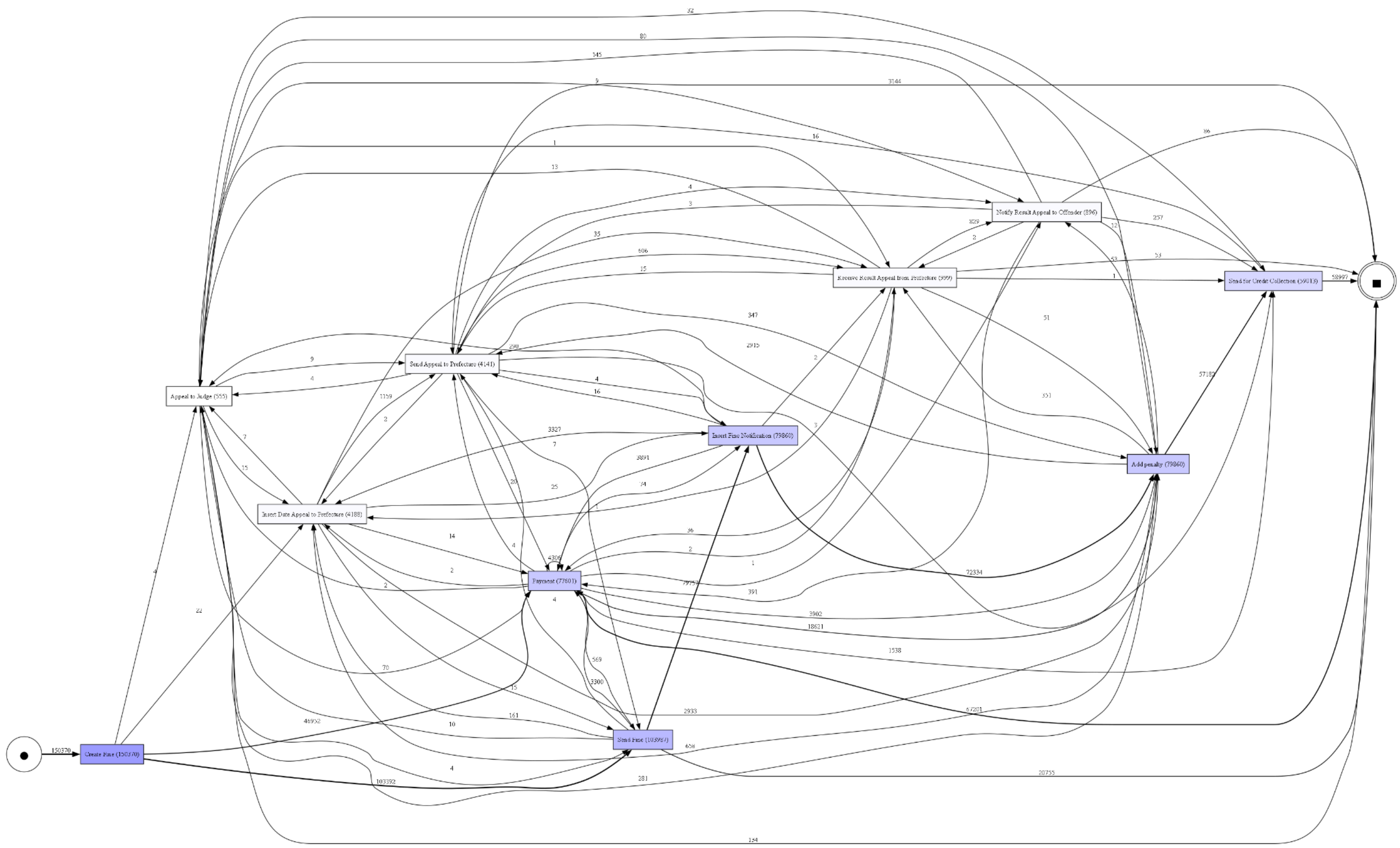
3.1 Directly-Follows Graph (DFG)

The Directly-Follows Graph provides a frequency-based visualization of the relationships between activities. Nodes represent activities, while directed edges indicate that one activity directly follows another in at least one trace. Edge weights reflect how often the relation occurs.

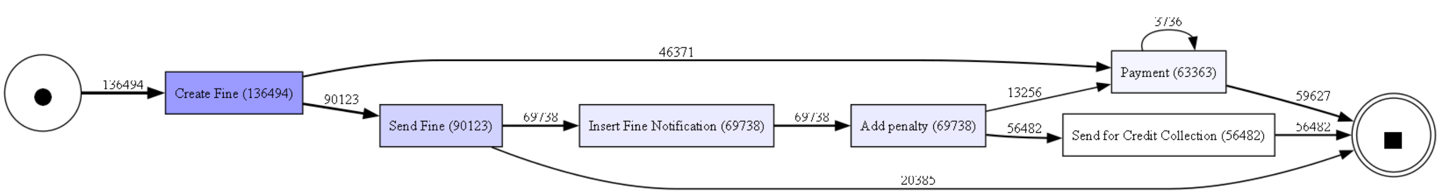
The DFG offers an intuitive overview of:

- The most frequent transitions between activities
- Main paths and alternative flows
- Potential loops and parallel patterns

Although it is not a formal execution model, the DFG is useful for quickly understanding the structure and complexity of the process before applying discovery algorithms.



DFG - Raw Log



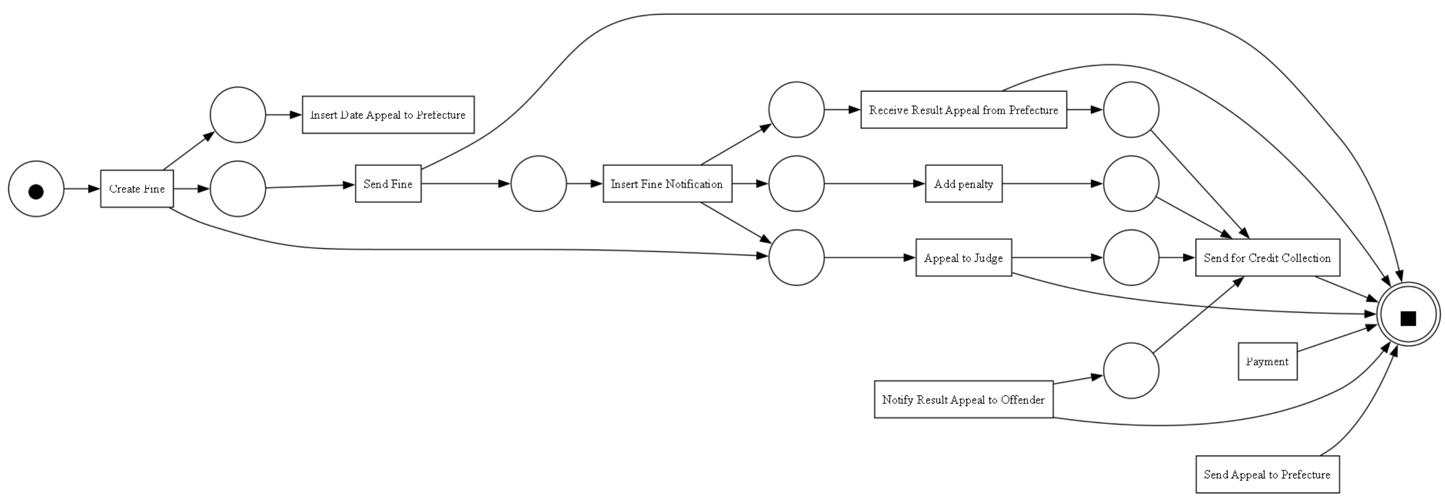
DFG - Filtered Log

3.2 Alpha Miner

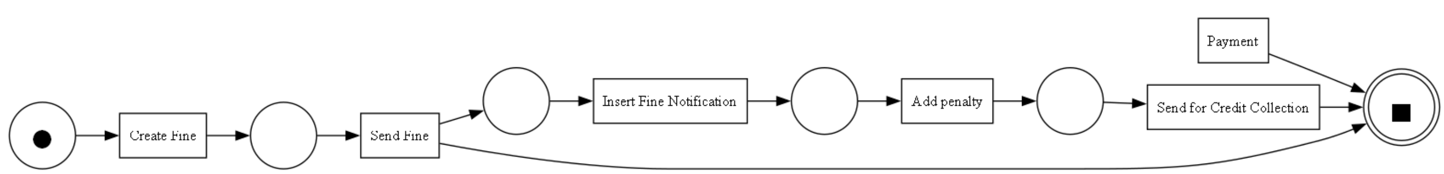
Alpha Miner is one of the earliest process discovery algorithms. It infers causal relations between activities based on ordering patterns in the event log. From these relations, it constructs a Petri net describing sequences, choices, and simple parallelism.
Characteristics:

- Conceptually simple and easy to interpret
- Sensitive to noise and infrequent behavior
- Limited handling of complex constructs

In this project, Alpha Miner is used primarily as a baseline technique for comparison with more robust algorithms.



Alpha Miner - Raw Log



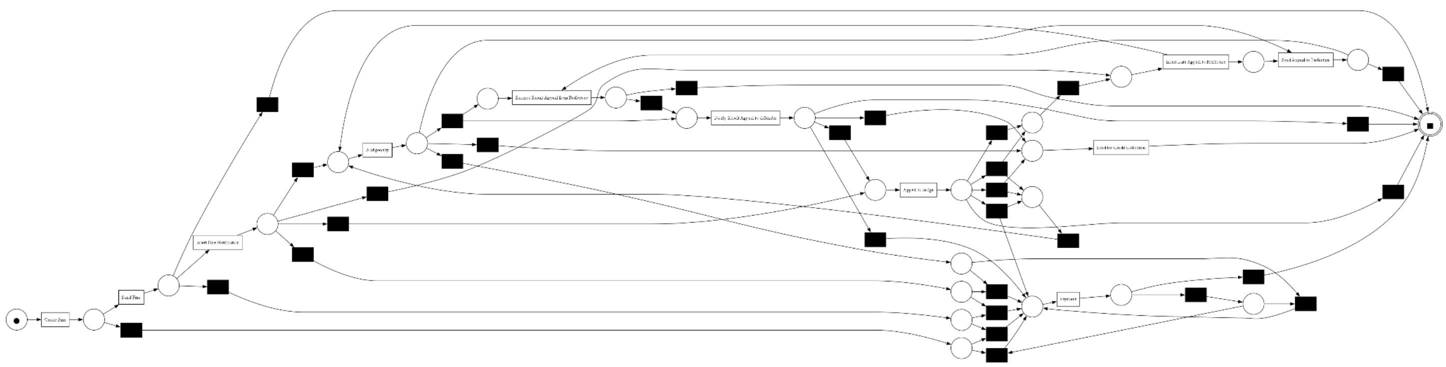
Alpha Miner - Filtered Log

3.3 Heuristics Miner

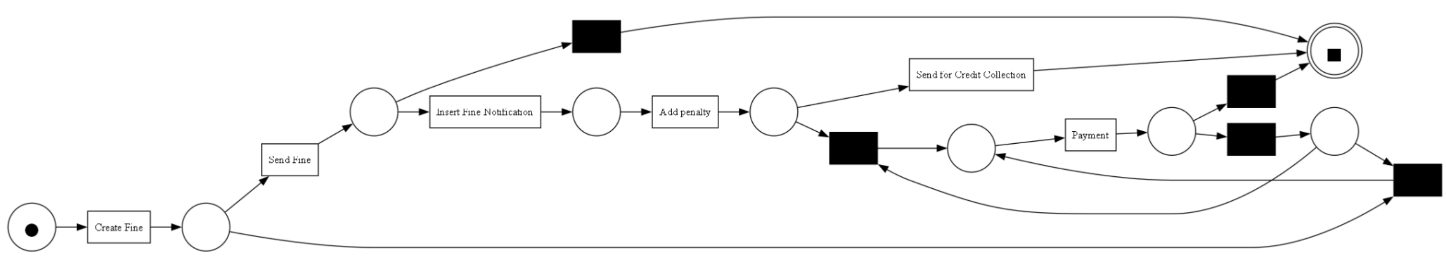
Heuristics Miner extends the Alpha approach by incorporating frequency information. Instead of relying purely on ordering relations, it evaluates dependency measures between activities, allowing it to filter out weak or noisy connections.
Characteristics:

- More robust to noise
- Handles short loops better than Alpha Miner
- Produces models reflecting dominant behavior

This algorithm is particularly suitable for real-life logs where variability and noise are present.



Heuristics Miner - Raw Log



Heuristics Miner - Filtered Log

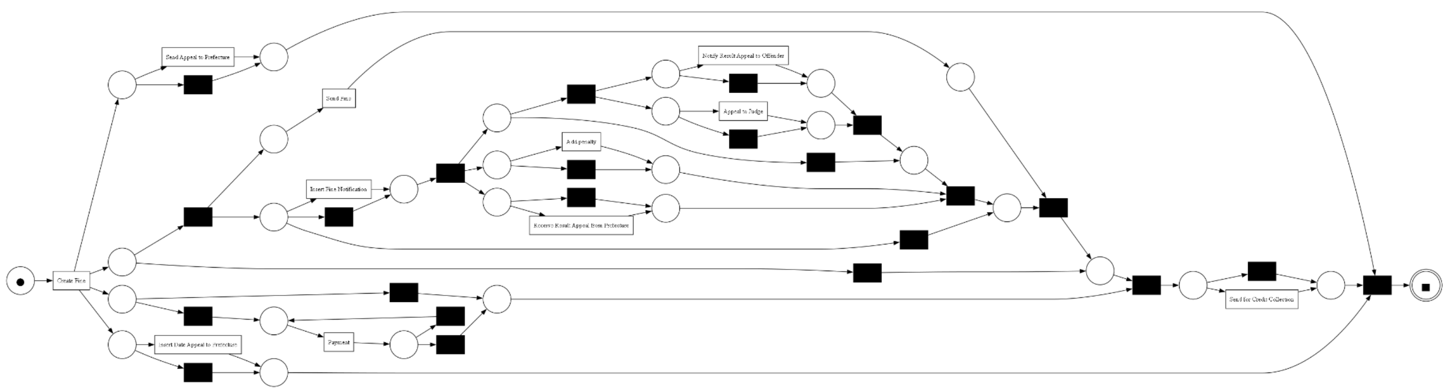
3.4 Inductive Miner

Inductive Miner follows a divide-and-conquer strategy. It recursively partitions the event log into subsets representing structured behavioral patterns (sequence, parallelism, choice, loops), and then constructs a sound process model.

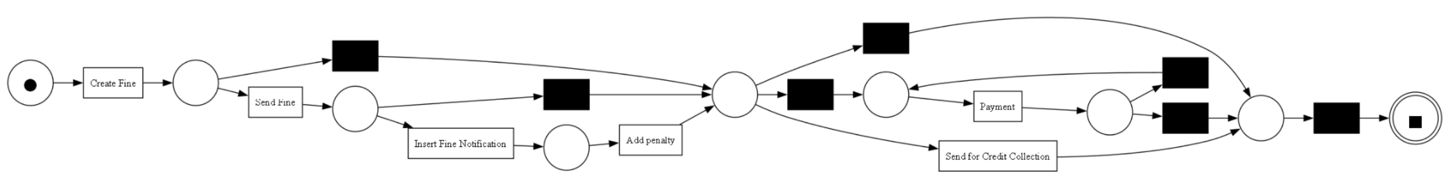
Characteristics:

- Guarantees sound Petri nets
- Robust to noise (depending on variant)
- Produces structured and often simpler models

Due to its stability and model quality, Inductive Miner frequently serves as a strong candidate for the final selected model.



Inductive Miner - Raw Log



Inductive Miner - Filtered Log

4 Model Evaluation

This section describes how the discovered process models are evaluated and compared. The goal of the evaluation is to assess the quality of the models generated by the three discovery algorithms and to support the selection of the most appropriate one for analysis.

All models have been evaluated against the raw event log using four quality metrics: fitness, precision, generalization, and simplicity. This evaluation strategy allows a consistent comparison of how models derived from different log configurations reproduce the full observed behavior.

4.1 Evaluation Metrics

Four standard quality metrics are used to evaluate the discovered Petri net models. These metrics are commonly adopted in process mining to capture different aspects of model quality.

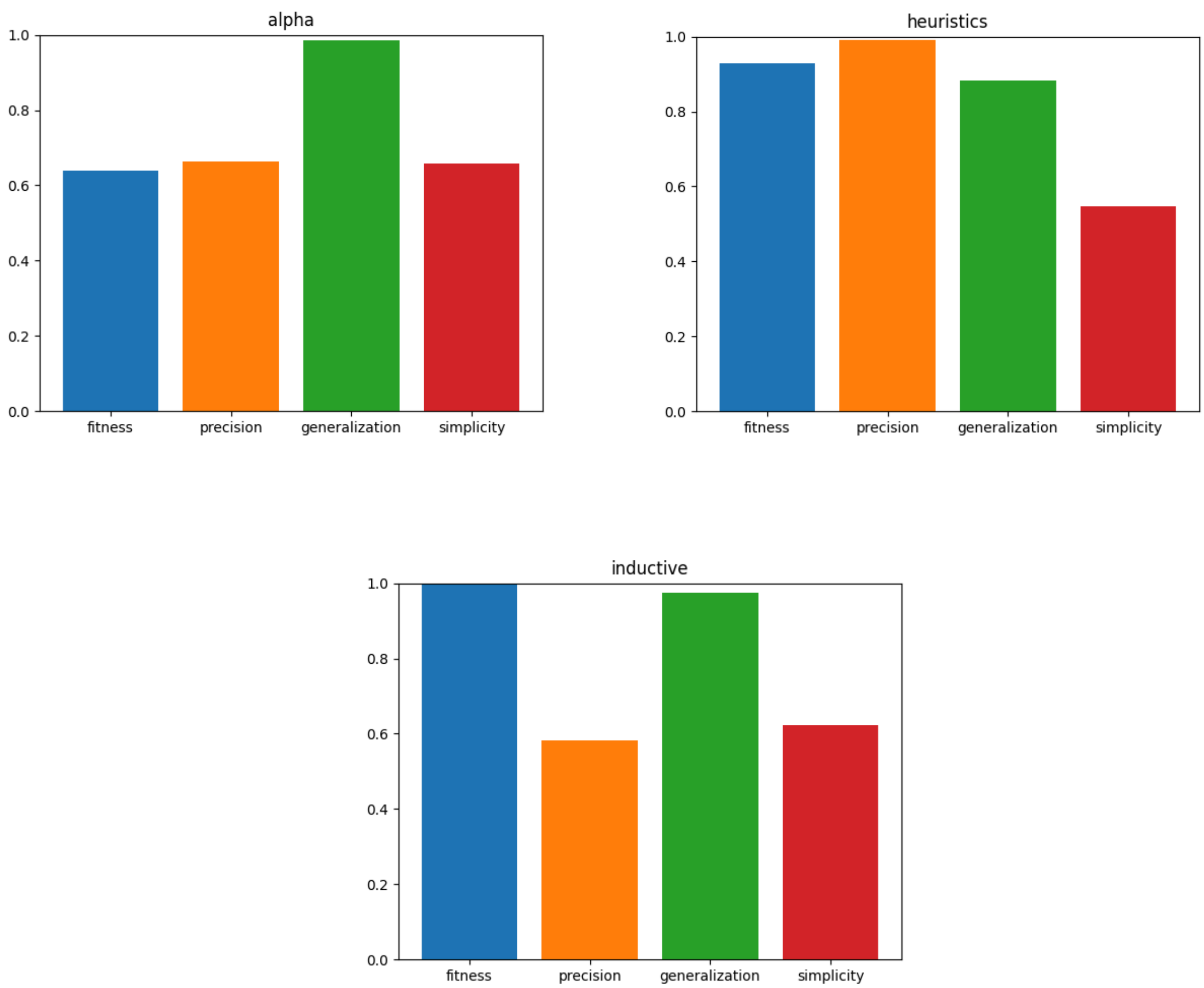
- ***Fitness*** → measures the extent to which a process model can reproduce the behavior recorded in the event log. A model with high fitness is able to replay most of the observed traces, while a model with perfect fitness can reproduce every trace from start to end. Low fitness indicates that relevant behavior present in the log cannot be explained by the model.
- ***Precision***: measures how much additional behavior is allowed by the model compared to what is observed in the event log. A model with high precision restricts its behavior closely to the recorded traces, while a model with low precision allows many behaviors that never occur in the log.
- ***Generalization*** → evaluates the ability of a model to represent not only the observed behavior, but also plausible future behavior that is not explicitly present in the event log. A model with poor generalization is too specific and tends to overfit the data, capturing noise instead of the underlying process structure. Generalization is estimated by analyzing how evenly different parts of the model are used during replay.
- ***Simplicity*** → reflects how easy a process model is to understand and interpret. A simple model avoids unnecessary structural complexity and provides a clear representation of the process. Simplicity is typically assessed through structural properties of the model, such as size and connectivity. Simpler models are generally preferred, provided that fitness and precision remain acceptable.

All metrics are computed using PM4Py and are normalized in the range [0,1], where higher values indicate better quality.

4.2 Raw Log Results

The models discovered from the raw log reflect the ability of each algorithm to capture the complete behavioral variability of the dataset.

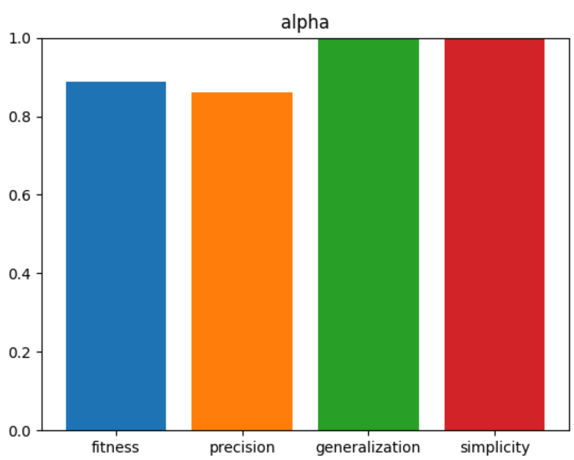
Alpha Miner achieves relatively low fitness (0.638), indicating that the discovered model is unable to replay a significant portion of the traces. Precision (0.662) is moderate, while generalization is very high (0.986), suggesting that the model allows a large amount of additional behavior. Simplicity (0.657) is acceptable but not particularly strong.
Heuristics Miner produces a model with high fitness (0.929) and extremely high precision (0.991). This indicates that the model closely matches the recorded traces while restricting additional behavior. Generalization (0.883) is lower than Alpha and Inductive, and simplicity (0.548) is the lowest among the three, reflecting higher structural complexity.
Inductive Miner achieves perfect fitness (1.000), meaning that the model can replay all traces in the raw log. Precision (0.582) is lower, indicating that the model allows additional behavior. Generalization (0.975) is high, and simplicity (0.624) remains balanced.
Overall, Inductive Miner provides the highest behavioral coverage, while Heuristics Miner provides the most precise model.

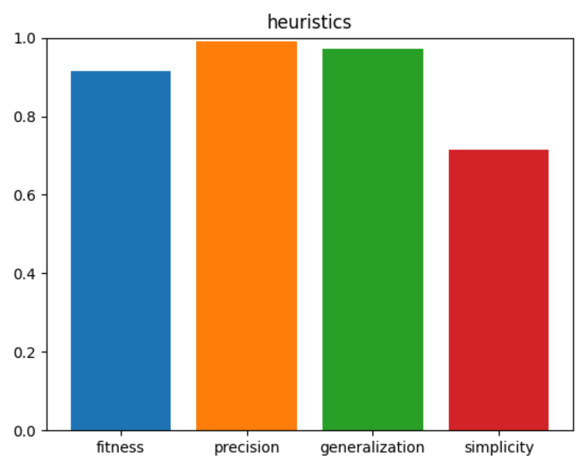


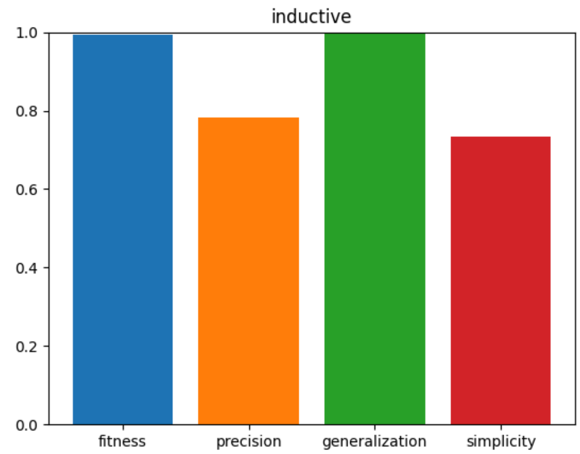
Metrics of the Raw Log

4.3 Filtered Log Results

Models discovered from the filtered log are evaluated on the raw log to assess how a simplified representation of dominant behavior explains the full dataset.
Alpha Miner shows a substantial improvement compared to its raw-log counterpart. Fitness increases from 0.638 to 0.887, and precision rises from 0.662 to 0.860. Generalization becomes nearly perfect (0.997), while simplicity reaches the maximum value (1.000). This indicates that filtering removes problematic relations and allows Alpha Miner to construct a cleaner and more stable process structure.
Heuristics Miner maintains strong performance after filtering. Fitness slightly decreases (0.929 → 0.914), while precision remains extremely high (≈0.990). Generalization improves significantly (0.883 → 0.971), and simplicity increases (0.548 → 0.714), indicating a reduction in structural complexity. ***Inductive Miner*** preserves very high fitness (1.000 → 0.994) with a clear improvement in precision (0.582 → 0.782). Generalization remains very high (0.995), and simplicity increases (0.733). This shows that filtering produces a simpler model while largely preserving the ability to replay the full log.
An important observation is that fitness decreases only marginally for Heuristics and Inductive Miner (approximately 0.01). Despite the filtered models being discovered from a simplified log, they remain highly compatible with the raw log. This suggests that the dominant variants retained during filtering capture most of the behavioral structure of the process, while rare variants contribute limited impact on the aggregate replay metric.







Metrics of the Filtered Log

4.4 Comparison and Model Selection

Comparing raw-log and filtered-log models reveals different algorithmic behaviors and trade-offs. ***Alpha Miner*** benefits significantly from filtering. The filtered Alpha model achieves much higher fitness, precision, and simplicity, indicating that Alpha Miner is strongly affected by noise and variability in the raw log. However, even after improvement, its precision and fitness remain below those achieved by the other algorithms.
Heuristics Miner consistently delivers excellent precision (≈0.99) and strong fitness in both configurations. Filtering mainly improves generalization and simplicity, reducing structural complexity without substantially affecting behavioral accuracy.
Inductive Miner provides the highest fitness overall. The raw-log model perfectly reproduces the log but allows additional behavior (lower precision). The filtered-log model slightly reduces fitness while improving precision and simplicity, resulting in a more balanced and interpretable model.

Considering the combined evaluation, each algorithm shows different strengths. Heuristics Miner achieves the highest precision, indicating a strong restriction of additional behavior. Inductive Miner achieves the highest fitness and strong generalization, ensuring that the observed traces can be reproduced reliably. The models discovered from the filtered log exhibit increased simplicity and improved precision, but they represent only the dominant variants retained during filtering. As stated earlier, the filtered log is intended to provide a clearer view of common scenarios rather than a complete representation of the process. For this reason, the ***Heuristics Miner*** model discovered on the raw log is selected as the preferred model for this project. It offers the most balanced performance on the raw dataset (fitness and precision both above 0.9), even though simplicity is lower (0.548), reflecting a more complex structure. The filtered-log models are used only as a complementary "common scenarios" view to improve interpretability, not as the main representation of the full process variability.

algorithm	raw_fitness	raw_precision	raw_generalization	raw_simplicity
alpha	0.637766272	0.662354432	0.985550039	0.657142857
heuristics	0.928550211	0.991362508	0.882981635	0.548387097
inductive	1	0.582252588	0.975259411	0.623762376

Metrics of the Raw Log

algorithm	filtered_fitness	filtered_precision	filtered_generalization	filtered_simplicity
alpha	0.887219147	0.860355643	0.996589436	1
heuristics	0.913796207	0.989961482	0.970778732	0.714285714
inductive	0.99441609	0.782271164	0.995124563	0.733333333

Metrics of the Filtered Log

5 LLM-Based Reasoning

To complement the quantitative process mining analysis, the dashboard integrates a Large Language Model (LLM) accessed through the Groq API. The LLM is used as a decision-support component to automatically interpret metrics, summarize results, and assist user interaction.

5.1 AI Process Report and Interactive Q&A

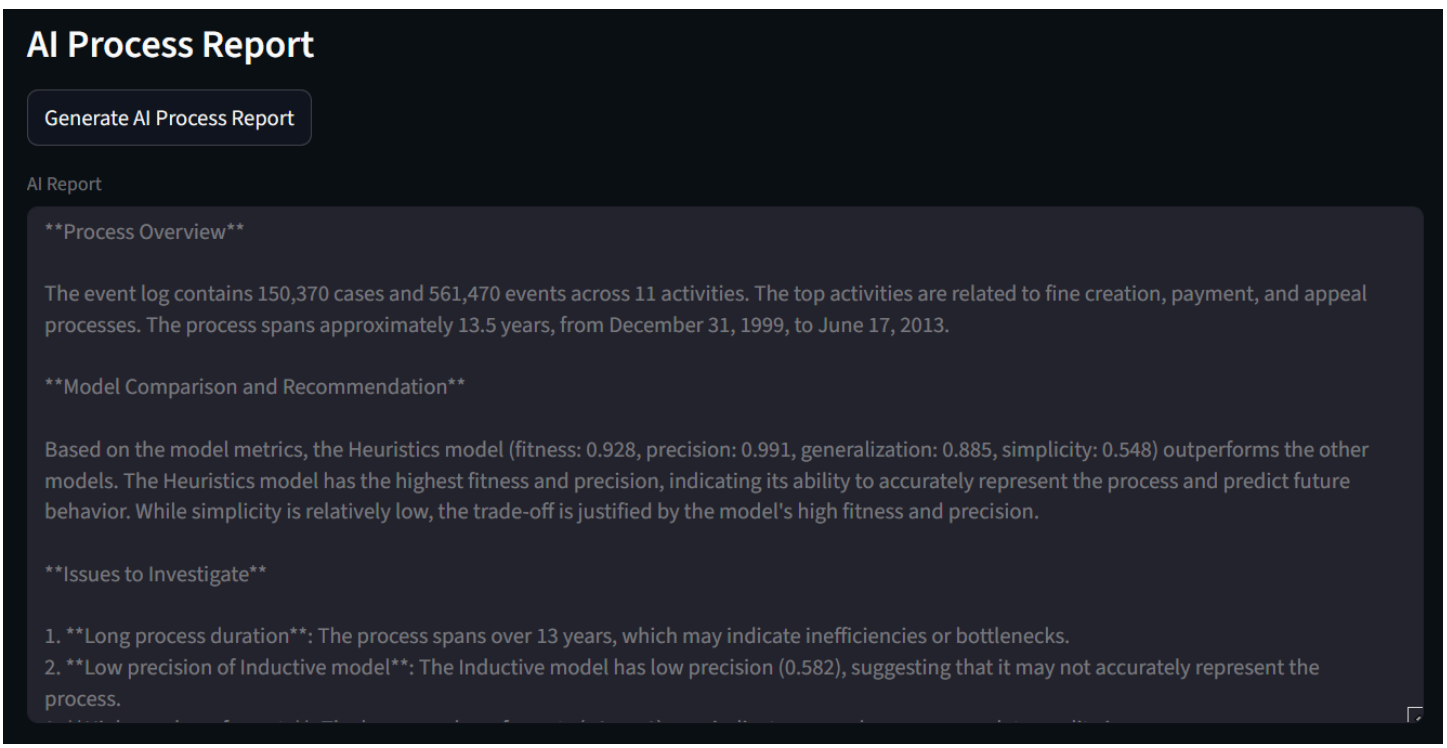
The AI Process Report generates a short textual analysis based on:

- Event log summary (cases, events, activities, time range)
- Model evaluation metrics (fitness, precision, generalization, simplicity)

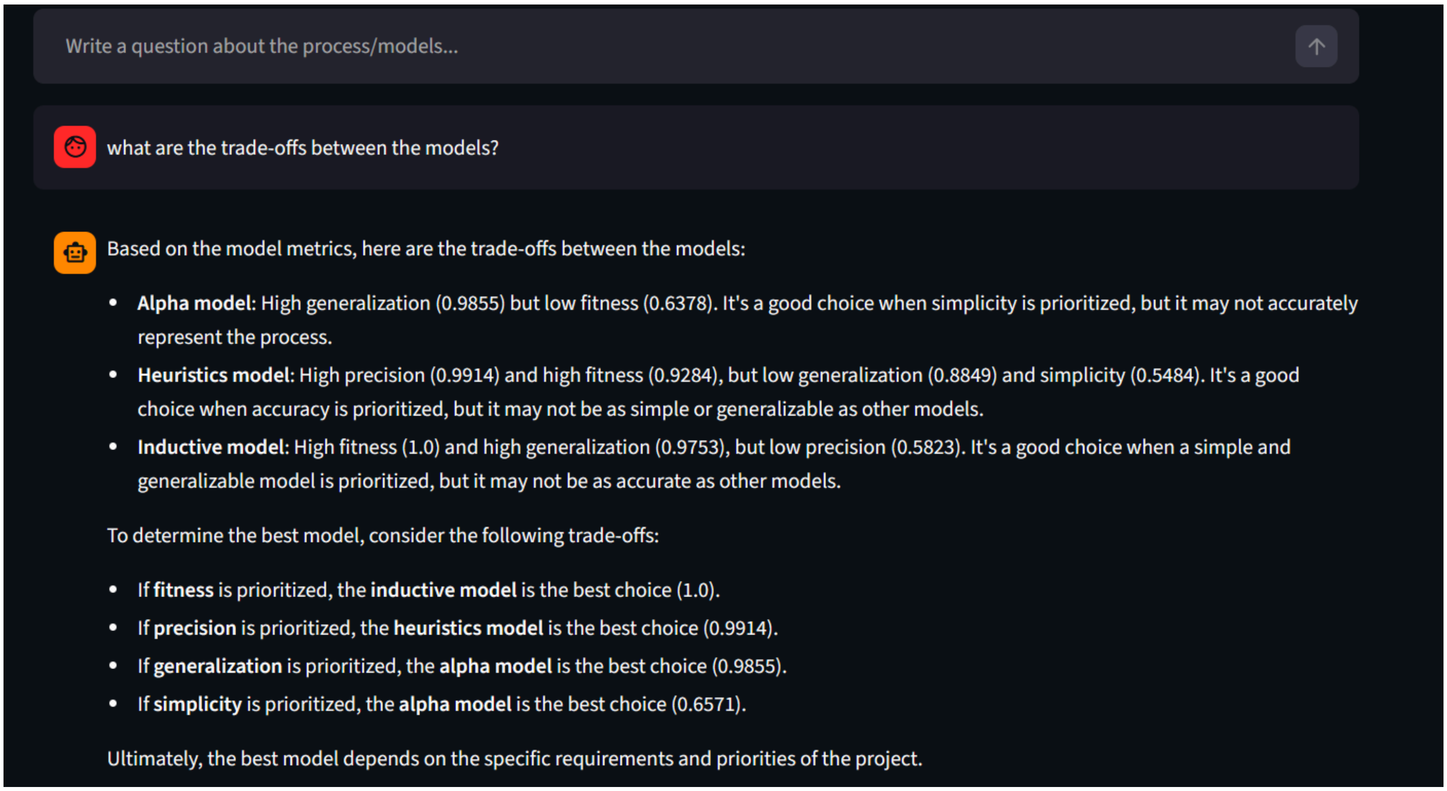
When triggered, the dashboard sends these elements to the LLM with a constrained prompt requesting:

- High-level process description
- Best model recommendation with justification
- Issues to investigate
- Suggested improvement actions

The model is explicitly instructed not to invent values and to rely only on the provided metrics. This feature transforms numerical results into an interpretable managerial-style report.



The dashboard also provides a chat interface where users can ask questions about process behavior, model differences, metric trade-offs, etc…
Each user query is combined with the same contextual information (log summary and metrics) and forwarded to the LLM. The responses are guided by rules enforcing conciseness, metric-based reasoning, and avoidance of unsupported claims. This component enables exploratory analysis without manually inspecting all metrics and visualizations.



5.2 Root Cause Analysis on Anomalies

A simple anomaly detection mechanism identifies cases with unusually long durations. Case durations are computed as the difference between the first and last timestamp per case. Cases exceeding the 95th percentile are labeled as anomalous.
For a sample of anomalous cases, the dashboard requests the LLM to:

- Suggest plausible explanations
- Indicate potential causes
- Propose verification checks

The LLM does not perform statistical analysis but provides qualitative reasoning that supports hypothesis generation. This feature illustrates how language models can assist interpretation tasks alongside traditional process mining techniques.

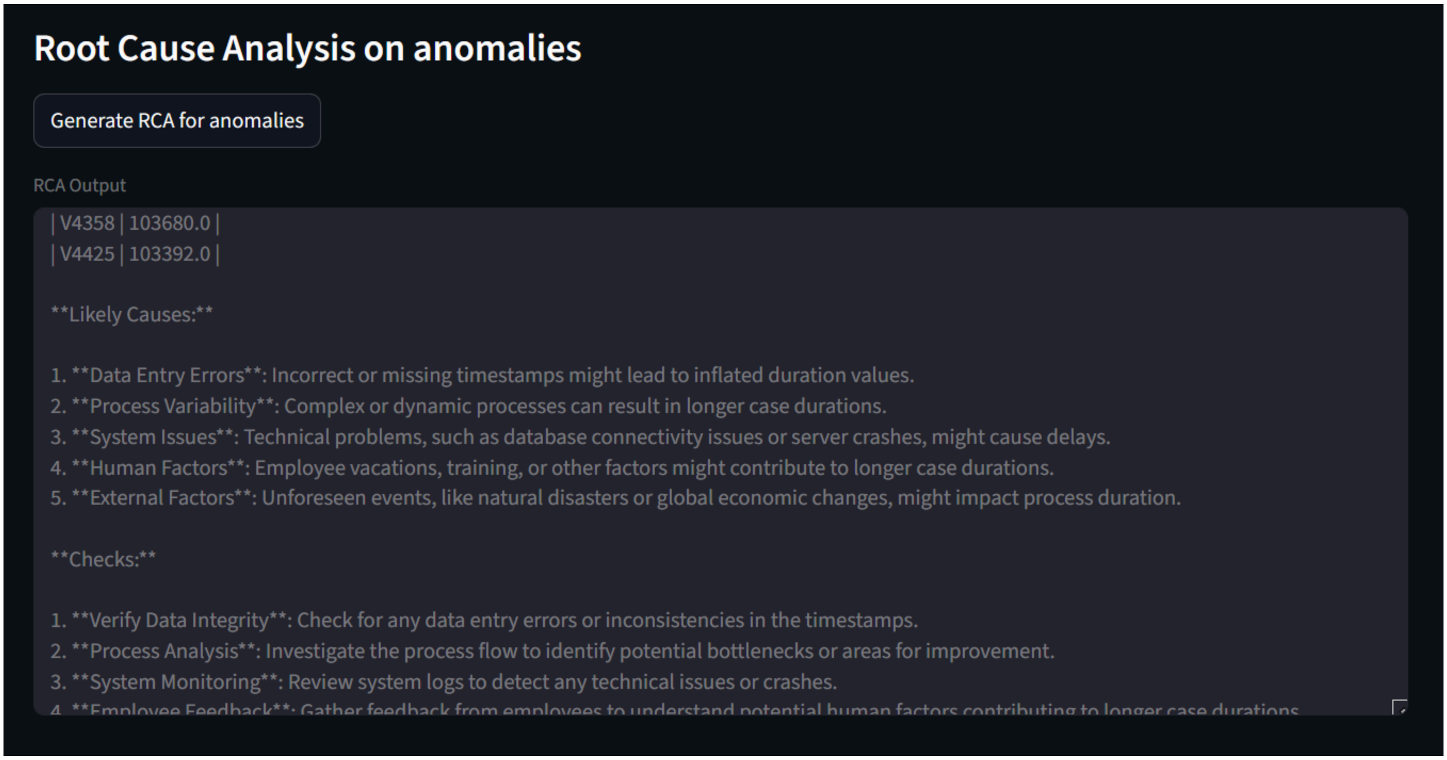
Anomaly detection (case duration)

Compute / Refresh anomalies

Threshold (95th percentile): **23232.00 hours**

Anomalous cases: **7435**

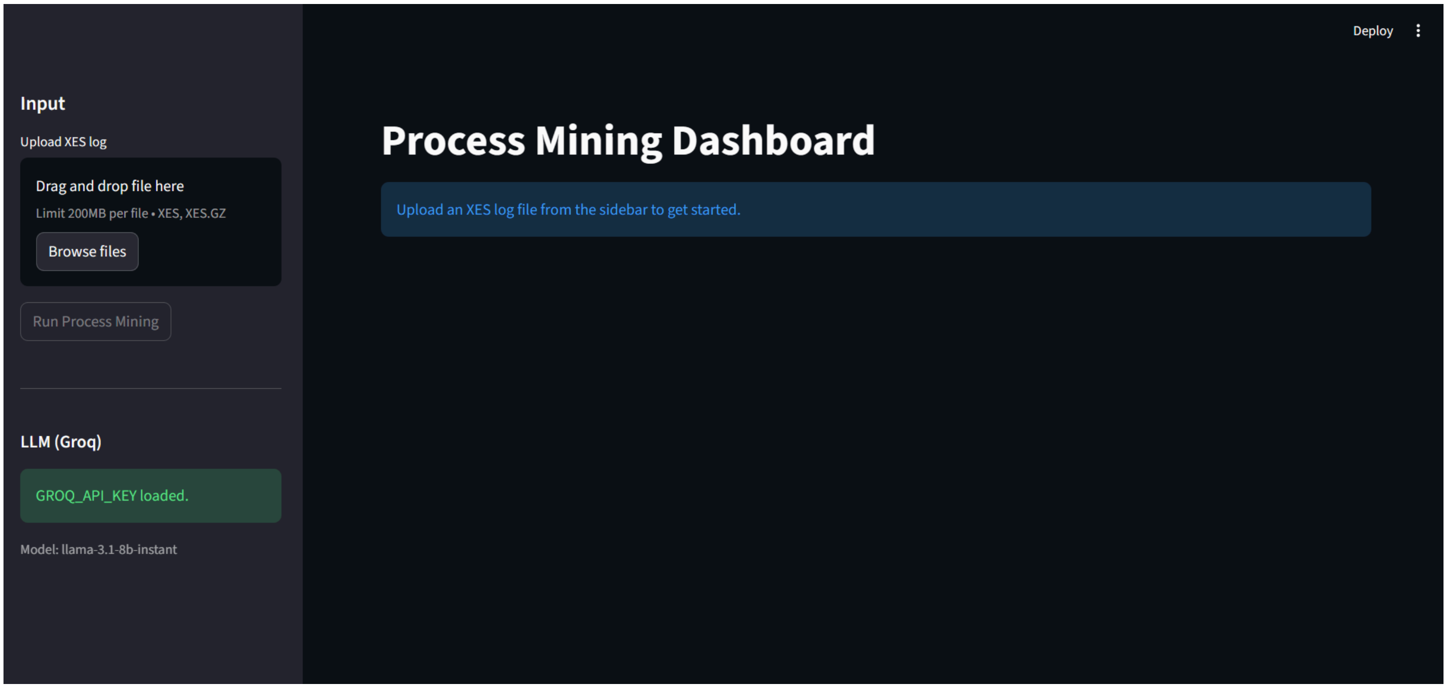
case:concept:name	min	max	duration_hours
V3904	2000-01-04 23:00:00+00:00	2011-12-24 23:00:00+00:00	104928
N22795	2000-01-24 23:00:00+00:00	2011-12-24 23:00:00+00:00	104448
V4290	2000-02-04 23:00:00+00:00	2011-12-24 23:00:00+00:00	104184
S38092	2000-02-06 23:00:00+00:00	2011-12-24 23:00:00+00:00	104136
V4358	2000-02-25 23:00:00+00:00	2011-12-24 23:00:00+00:00	103680
V4425	2000-03-08 23:00:00+00:00	2011-12-24 23:00:00+00:00	103392
V5141	2000-06-07 22:00:00+00:00	2011-12-24 23:00:00+00:00	101209
V5285	2000-06-12 22:00:00+00:00	2011-12-24 23:00:00+00:00	101089
V5624	2000-06-22 22:00:00+00:00	2011-12-24 23:00:00+00:00	100849
V6005	2000-07-18 22:00:00+00:00	2011-12-24 23:00:00+00:00	100225



Overall, the LLM integration serves as an interpretative layer that enhances readability, supports interaction, and aids exploratory reasoning over process mining results.

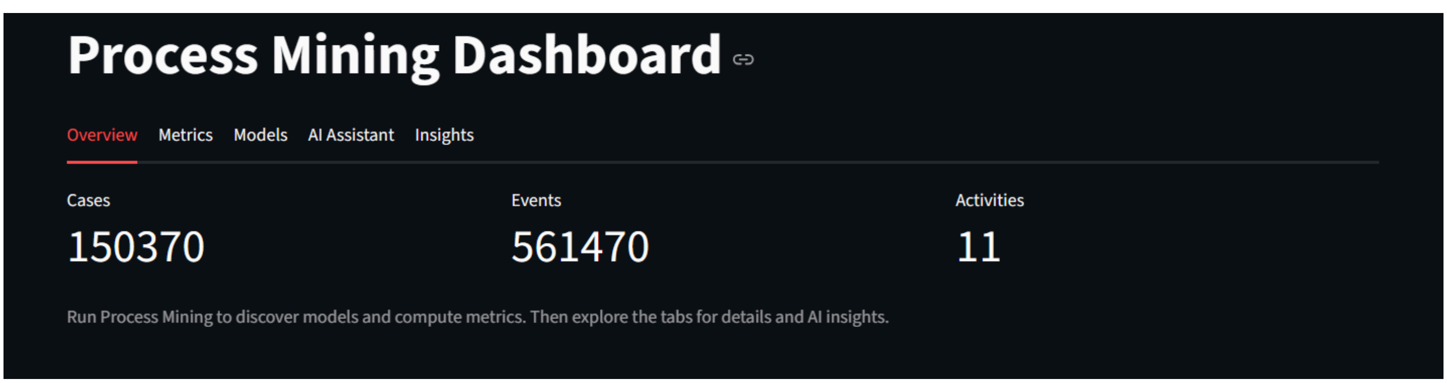
6 Dashboard Implementation

A Streamlit-based dashboard was developed to provide an interactive interface for exploring the event log, visualizing discovered models, inspecting evaluation metrics, and integrating LLM-based reasoning. The dashboard is designed as a lightweight analytical tool that combines process mining outputs with user-friendly interaction.



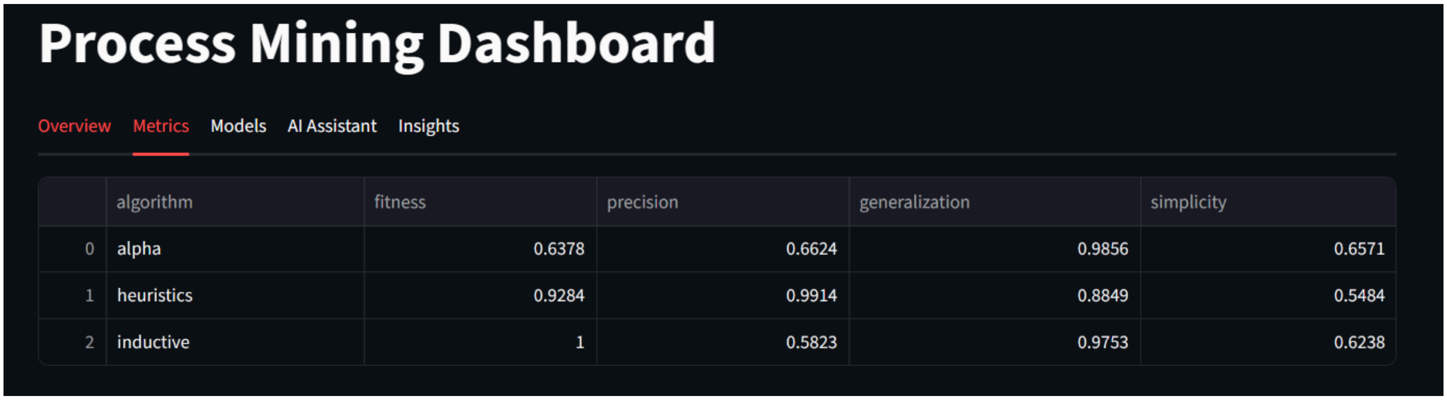
6.1 Overview Tab

The Overview tab presents basic statistics extracted from the uploaded event log. These include the number of cases, number of events, and number of distinct activities. This section offers an immediate understanding of the dataset size and complexity before running discovery and evaluation procedures.



6.2 Metrics Tab

The Metrics tab displays a structured table containing the evaluation results for each discovery algorithm. Fitness, precision, generalization, and simplicity are shown side by side, allowing direct comparison of model quality. This view supports analytical interpretation without requiring manual inspection of individual charts.


Process Mining Dashboard

Overview Metrics Models AI Assistant Insights

	algorithm	fitness	precision	generalization	simplicity
0	alpha	0.6378	0.6624	0.9856	0.6571
1	heuristics	0.9284	0.9914	0.8849	0.5484
2	inductive	1	0.5823	0.9753	0.6238

6.3 Models Tab

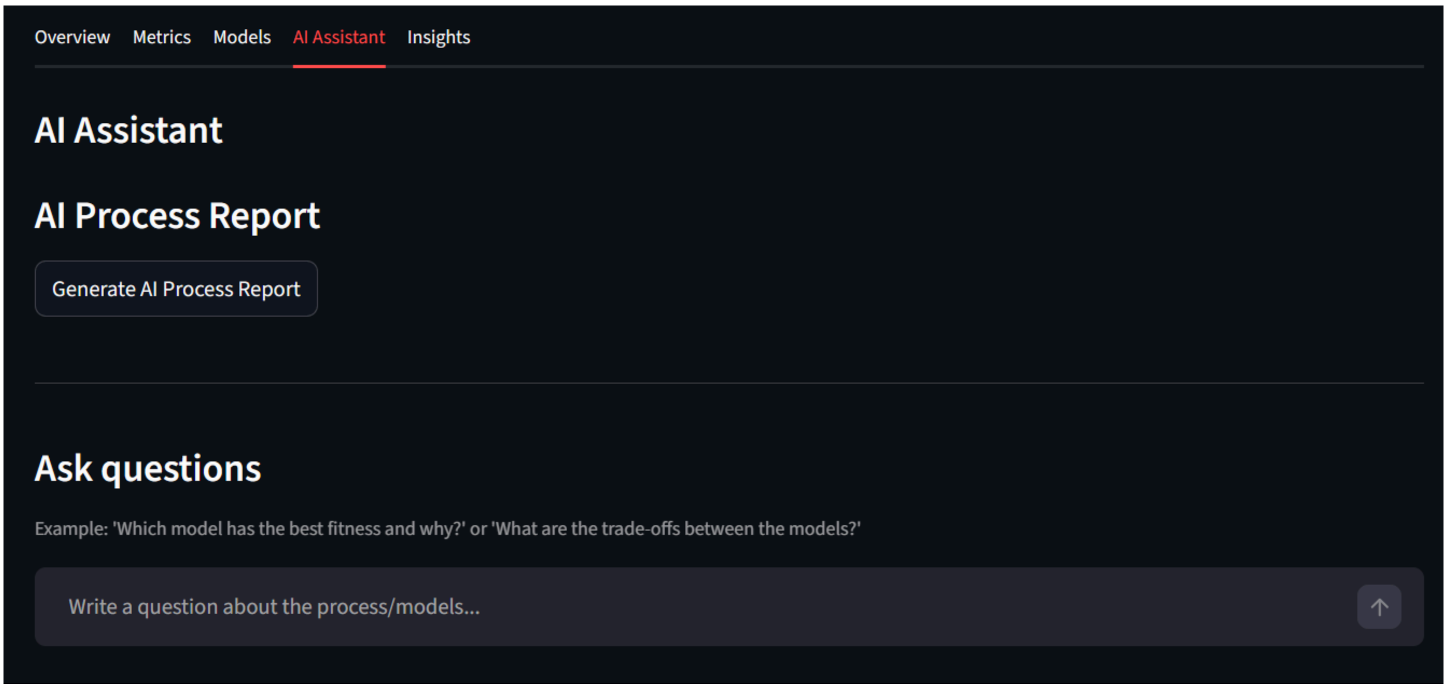
The Models tab visualizes the Petri nets discovered by the three algorithms (Alpha Miner, Heuristics Miner, Inductive Miner). For each model, the dashboard shows both the Petri net visualization and the corresponding metric bar chart. This tab links quantitative evaluation with structural interpretation, helping assess model complexity and readability.

6.4 AI Assistant Tab

The AI Assistant tab integrates the Groq-based LLM.
Two main features are provided:

- AI Process Report, which generates a concise textual summary and recommendations based on log statistics and metrics
- Interactive Q&A, which allows users to ask questions about models, metrics, and trade-offs

This component adds an interpretative layer to the dashboard.

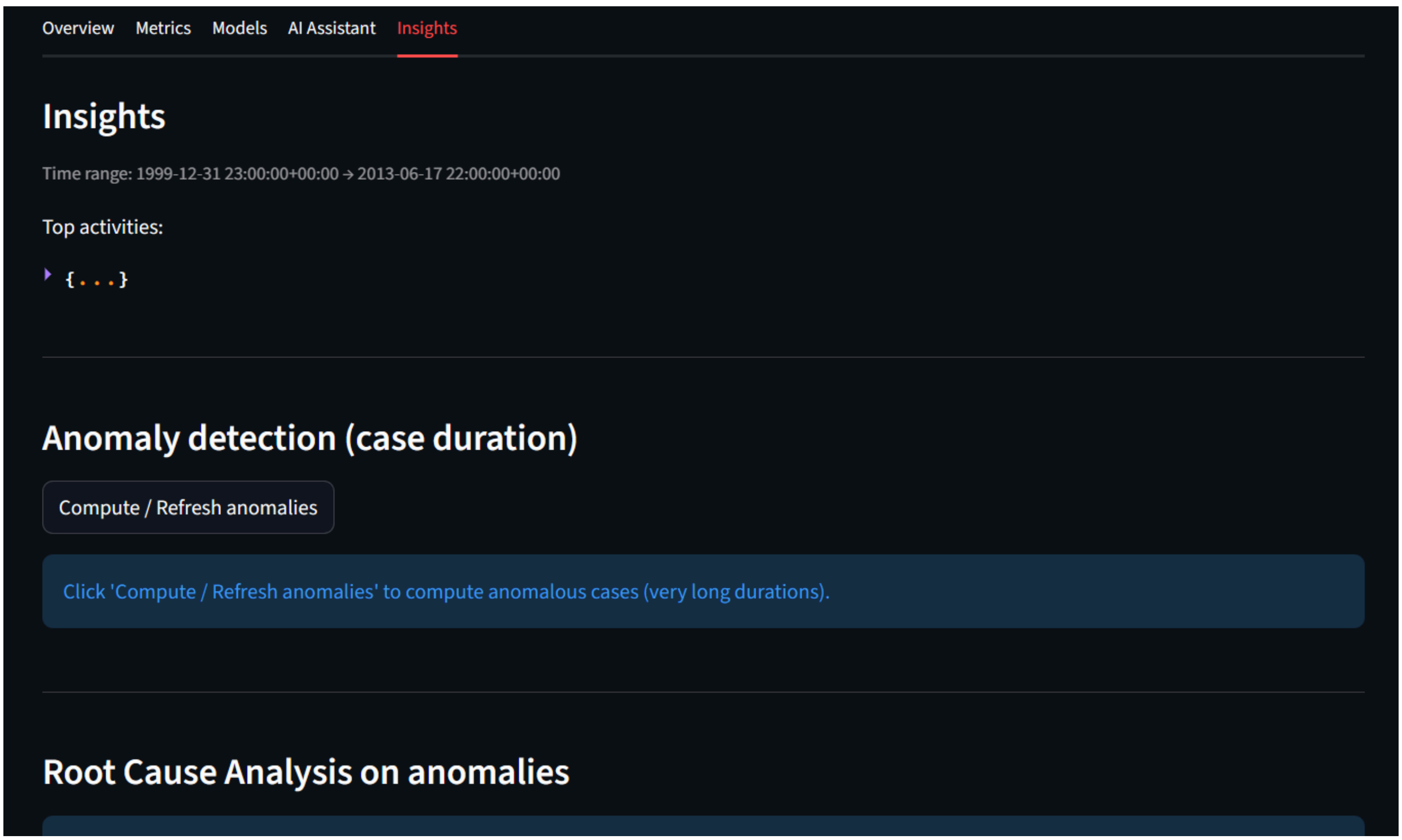


6.5 Insights Tab

The Insights tab includes:

- Summary information (time range, top activities)
- Case duration analysis
- Detection of anomalous cases based on the 95th percentile of duration

Detected anomalies are displayed in a table. An optional Root Cause Analysis function queries the LLM to provide qualitative explanations and investigation suggestions for sampled anomalous cases.



7 Conclusion

This project applies a complete process mining workflow to a real event log. Starting from the Road Traffic Fine Management Process dataset, the analysis includes log preprocessing and filtering, process discovery using three algorithms, model evaluation through standard metrics, and result exploration through an interactive dashboard. The comparison of discovery techniques highlights different trade-offs. Heuristics Miner is selected as the best model for the raw log because it combines high fitness and very high precision, closely reflecting the observed behavior. Inductive Miner achieves perfect fitness but allows more behavior than observed. Log filtering is used to obtain a clearer view of common execution scenarios and to improve model interpretability, without replacing the analysis of the full process. The dashboard and the LLM-based components support result interpretation by providing automated reports, interactive questions, and simple anomaly explanations. While the evaluation and anomaly detection are intentionally lightweight, they effectively illustrate the main ideas of process mining. Future improvements could include the use of more advanced conformance metrics, richer anomaly detection techniques, and deeper predictive or prescriptive analysis. Overall, the project demonstrates how process mining methods can be applied end to end to analyze, compare, and interpret real process behavior in a clear and practical way.