Formal Method Project

PROCESS MINING PROJECT WITH AI SUPPORT

GADALETA TIZIANO (860987)

Table of contents

Abstract

Process Mining is a data-driven approach aimed at improving Business Process Management (BPM) by using IT system event logs to reveal how processes actually function, identifying bottlenecks and deviations.

Traditional Business Process Management often is based on subjective and manual inputs (such as interviews and workshops) for idealized models, whereas Process Mining acts as a bridge, adding fact-based quantitative insights (discovery, compliance, improvement) to the qualitative lifecycle of BPM design and management, thus transforming it from a mere hypothesis to a scientific approach for continuous improvement.

The case study for this project analyzes an online cosmetics e-commerce store.

The formal objectives of the project are the following:

- Event Log Preprocessing and Feature Engineering: event log management in CSV format from a cosmetics e-commerce site, including data cleaning, normalization, dimensionality reduction and filtering based on unique user sessions, to obtain high-quality logs for process analysis.
- Process Discovery and Model Extraction: application of process discovery algorithms (Alpha Miner, Heuristics Miner, Inductive Miner) for the automatic extraction of representative process models from preprocessed event logs.
- Process Model Evaluation and Comparison: evaluation and comparison of discovered process models through standard process mining metrics, including fitness, precision, generalization and simplicity, to analyze quality and trade-offs.
- Next-Event Prediction and Explainability: prediction of the next process event based on the sequences observed in the events log, accompanied by an explanatory analysis of the predictions to support the interpretability of the results.
- LLM-Driven Analytical Support and Reporting: integrating a Large Language Model via OpenRouter to generate strategic reports, explaining process mining results and creating a chatbot to support algorithm and metric analysis.

1. Dataset

1.1 Description

The project's case study focuses on analyzing user sessions on an online store (e-commerce) that sells cosmetics.
The dataset used was released on the Kaggle platform by Michael Kechinov in 2020 and contains behavior data for 5 months (Oct 2019 - Feb 2020) from a medium cosmetics online store. Each row in the file represents an event. All events are related to products and users. Each event is like many-to-many relation between products and users.

The dataset can be downloaded in CSV format (Comma-Separated Values), a text file format used to represent tabular data. Each row corresponds to a record and the values are separated by a delimiter, typically a comma or semicolon. The data within contains key information about user interactions on the platform, such as:

- Event time: when event is was happened;
- Event type: event type, such as adding a product to the cart, removing a product from the cart and selling a product;
- Product id: identifier of the product in question;
- Category ID: identifier of the product category;
- Category code: Category meaningful name;
- Brand: Brand name in lower case;
- Price: price of the product;
- User id: user identifier (permanent);
- User session: session identifier.

The dataset used in the case study is the CSV file of the February 2020 events log (the most recent among those present), with a total size of 488.8MB and a number of unique sessions (User sessions) equal to 931,669. Each row in the file corresponds to an event log produced within the web platform. The data set provides a useful overview for understanding the specific situation of the case study.

1.1 Pre-processing

Before beginning any analysis and process modeling activities, it is essential to process the dataset to adapt it to the case study.
This process aims to transform the raw event dataset, stored in CSV format, into a normalized event log suitable for subsequent analysis activities. Processing involves a pipeline of validation, filtering, temporal normalization and semantic column standardization.

As a first step, the dataset size was reduced. This was done to maintain the computational efficiency of the discovery algorithms, keeping in mind that this project is a case study and not a true corporate Process Mining activity. This first filtering phase was performed according to two specific criteria: the number of unique cases (User sessions) and the number of unique case logs. A limit of 10,000 unique cases with a minimum of two activities per session was set, thus generating a new CSV file with 118,237 rows. The decision to generate a new CSV was made to avoid modifying the original CSV with all the event logs, allowing for future modifications and new tests to be performed on a larger number of unique cases. This could be easily modified by changing the input parameters of the function that manages the entire pre-processing phase on the dataset. By modifying and choosing values other than the default ones for the maximum number of unique sessions and the minimum number of events per session, you can easily obtain a new, smaller or larger dataset, tailored to your objectives.

After this initial dataset screening, incomplete or unusable observations are removed by deleting rows without session identifiers and removing events with invalid or uninterpretable timestamps. This phase ensures the minimum integrity required for temporal and sequential analysis. The session identifier, user identifier, time and event type are predefined parameters and adapted to the case study CSV, but they can be easily initialized for other datasets in the function declaration.

The next step involves converting the time values into a standardized datetime format. This allows for chronological sorting of events, compatibility with other process analysis tools and consistent temporal comparisons. The temporal sorting is performed based on the timestamp and session identifier, thus ensuring a consistent timeline.

As a final normalization task, the CSV columns were renamed according to the XES standard, an XML-based standard for representing events log. This format is commonly used in Process Mining to describe traces, events and associated attributes; it supports extensibility and formal semantics. Renaming the fields makes the dataset interoperable with process analysis tools and frameworks, thus ensuring the generation of a new dataset that conforms to the semantic and structural conventions commonly adopted in event log standards. The renamed columns relate to the unique identifier of the session, the unique identifier of the actor generating the event, the time of the event and the type of event.

Once all normalization activities are completed, the new CSV format dataset will be saved in the “data” directory as processed_event_log.csv.

2. Process discovery

2.1 Introduction

Process discovery is one of the central techniques of process mining and consists of automatically reconstructing a process model using only data recorded in information systems.
Rather than starting from a theoretical or documented description of the operational flow, the model is created by observing the behavior traced in event log.
This approach allows for an objective representation of the processes actually executed, often revealing deviations, unexpected loops, or alternative paths that would otherwise be unknown in the formal specifications.

After CSV normalization, it was possible to apply different process discovery algorithms, each characterized by a different approach to model construction and a different balance between quality metrics: Alpha miner, Heuristic Miner and Inductive Miner.

Alpha Miner is one of the first algorithms developed for the discovery of processes and is conceptually based on the analysis of precedence relationships between tasks. It is a deterministic algorithm and its operation is based on the identification of recurring patterns in sequences of events and, in particular, on direct dependencies and causal relationships between pairs of activities.
The result is a model that describes the flow of actions, highlighting the presence of parallelisms, alternative choices or linear sequences. The algorithm exhibits limitations when the data contains noise, exceptions or unstructured behavior.

Heuristic Miner, on the other hand, was created to address the critical issues of real logs, which often include anomalies or insignificant sequences.
Unlike Alpha Miner, this algorithm also relies on the frequency of events to assess the relevance of relationships between activities. In this case, not all observed connections are given equal weight, as marginal or accidental paths can be excluded and recurrent ones take on greater importance in building the model.

Inductive Miner, on the other hand, is a modern algorithm based on the recursive decomposition of the log into simpler substructures, which are then recombined according to certain hierarchical patterns.
This method guarantees important formal properties, such as the absence of inconsistent blocks or cycles, creating structurally defined models. The result tends to be more orderly and interpretable, with a clear distinction between sequences, parallelisms and choices.

To evaluate the effectiveness of the process models generated by the three algorithms Alpha Miner, Heuristic Miner and Inductive Miner it is essential to consider various quality dimensions. These are quantified through four main metrics: fitness, precision, generalization and simplicity.

Fitness describes how much of observed behaviour is allowed by the model. A model with good fitness is able to replay most of the traces of the log. A model with perfect fitness is able to replay all the traces in the log from beginning to end.

Precision relates to how much of the behaviour allowed by the model is present in the log. Precision is high if the model does not allow for much more behaviour than recorded in the log. It is low, if the model generally allows for a lot of behaviour even if this is not part of the log. An algorithm with good precision tries to minimize. Therefore, a perfect log precision is not necessary in able to get a good process model. Precision can also be related to the notion of underfitting known from the context of data science. A model with poor precision is underfitting as it allows for behaviour that is very different to the behaviour seen in the log.

Generalization describes the likelihood that future behaviour will already be executable in the model. The notion of generalization is in some ways related to the notion of overfitting known from the field of data science. If a model is overfitting, it is likely to also support variations such as noise. Through this it gets too specific and therefore fails to truly capture the underlying structure of the data. So, a model with good generalization aims to prevent overfitting. One approach to measure generalization is to use alignment to see how frequently parts of the model are used. If all parts of the model are used frequently, most likely all future behaviour is captured and therefore generalization is high. If there are parts of the model that are infrequently used, it is more likely that there is behaviour that is not seen yet. This makes generalization low.

Simplicity relates to the first quality criteria. A model has good simplicity if it lacks complexity and is easy to understand for a human. Measuring simplicity on the other hand is not a trivial matter. In literature, there are many different approaches to measure this quality criteria. For example, some approaches take the size or the diameter of the model into account. Size can refer to the number of nodes in the model and diameter in this case refers to the length of the shortest path from a start node to an end node.

2.2 Application

To put into practice what has been seen in Process Discovery (i.e. the automatic discovery of process models starting from logs and the evaluation of their quality through fitness, precision, generalization and simplicity) it is necessary to rely on specialized software tools.

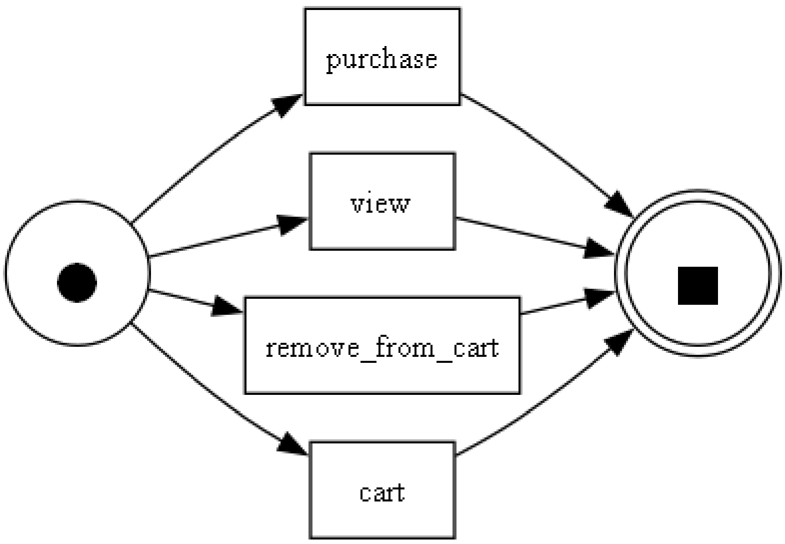
In this case study PM4Py, an open-source Python library specifically designed for Process Mining, was used. PM4Py was designed to bring advanced process analysis techniques directly into data science workflows, supporting major event log standards such as XES and CSV.
This library includes tools for cleaning, transforming and enriching data before analysis and is considered one of the best in its field.
It functions as a modular and extensible framework, similar to ProM and covers the main areas of process mining such as processes discovery, conformance checking, performances analysis and model enrichment. PM4Py also provides data structures and formal models, such as Petri nets, BPMN (Business Process Modeling Notation) and Process tree, to visualize the discovered processes.

Within the code, PM4Py is initially used to transform the pandas DataFrame into a PM4PY-compatible Event Log and, subsequently, to discover process patterns using the three algorithms: Alpha Miner, Heuristic Miner and Inductive Miner. More specifically, PM4Py is used in Alpha Miner to identify direct causal relationships between activities and return a Petri Net with start and end markings; in Heuristic Miner to manage log noise, filter out infrequent paths and produce the corresponding Petri Net (transformed from Heuristic Net to Petri Net for compliance with other standards); and in Inductive Miner to produce formally correct models (robust and deadlock-free) and generate the corresponding Petri Net (transformed from Process Tree to Petri Net). Other uses of PM4Py include generating images of discovered Petri Net in PNG format and evaluating the quality of the models with respect to the Event Log.

2.3 Results

The values obtained from the three metrics are as follows:

Alpha Miner



Fitness

0.156

Precision

1.000

Generalization

0.993

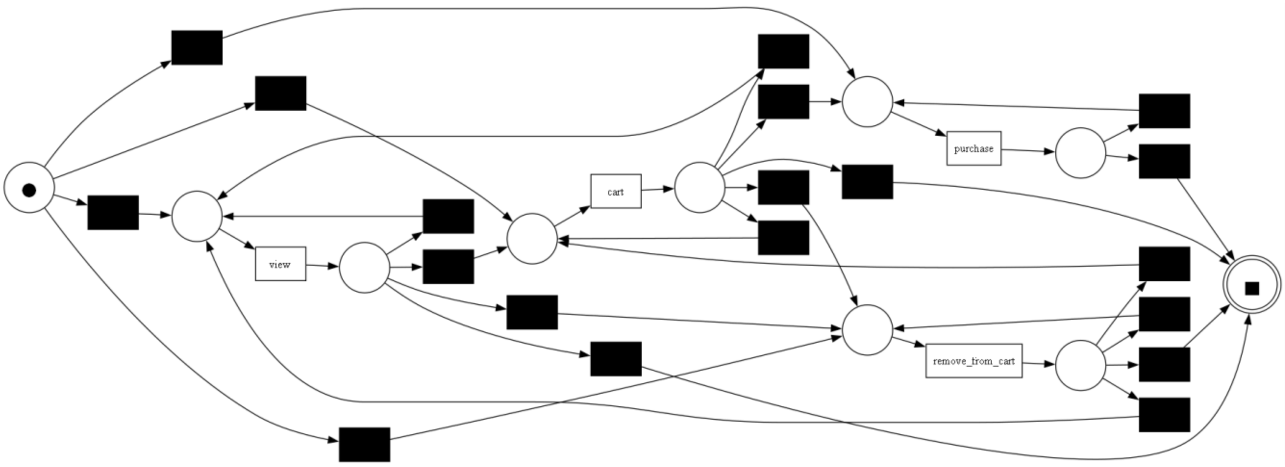
Simplicity

0.071

Alpha Miner Petri Net.

The model generated with Alpha Miner is extremely simple and rigid.
The precision is 1.000, meaning that the model only allows for the behaviors observed in the log, without admitting variations or alternative paths..
However, the very low fitness (0.156) highlights that the model is only able to reproduce a small portion of the real cases present in the event log. This is consistent with the nature of the Alpha Miner, as it tends to fail in the presence of repetitive behaviors, short loops, imperfectly structured alternative choices, typical noise of real logs of a product e-commerce site.
In the context of an online cosmetics store, where users can freely explore products, add and remove items from their carts multiple times and abandon the process, the Alpha model is far too simplified and, consequently, poorly representative of reality.

Heuristic Miner



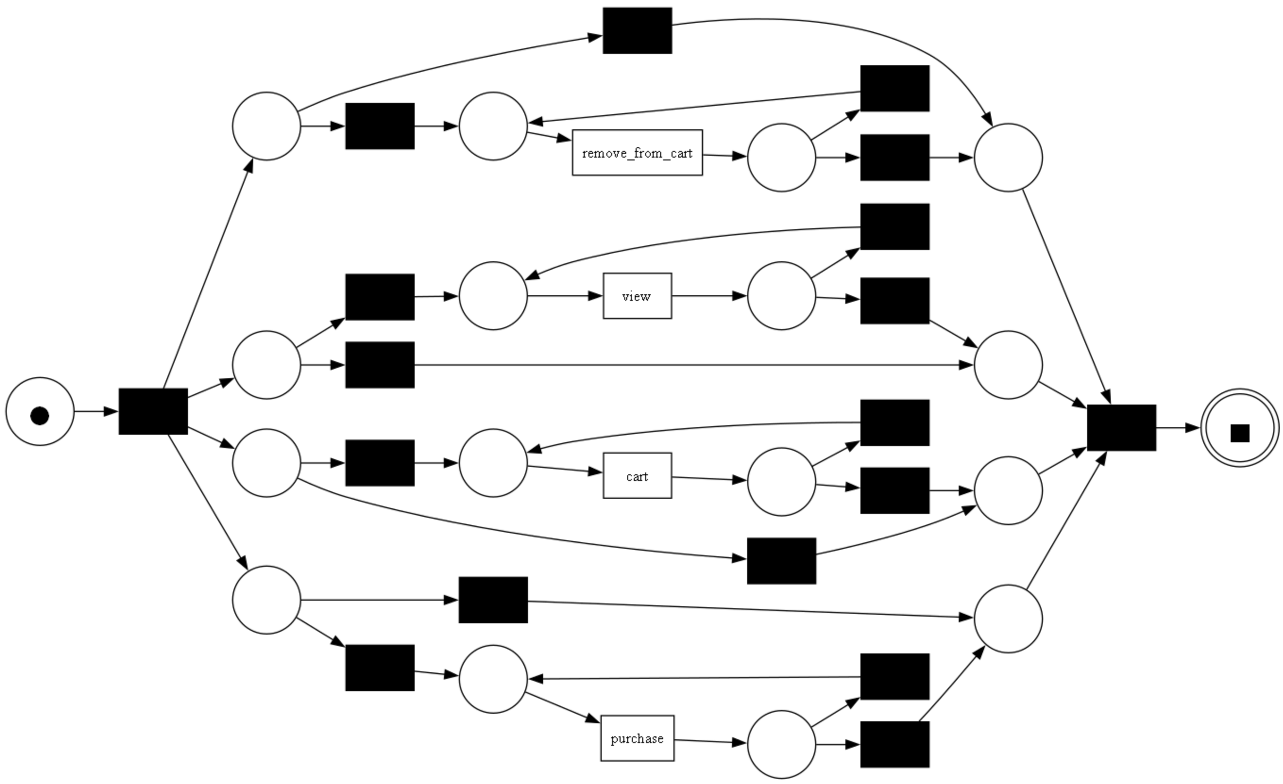
Heuristic Miner Petri Net.

Fitness	Precision	Generalization	Simplicity
0.998	0.645	0.978	0.013

The model obtained with Heuristic Miner shows an excellent balance between data accuracy and descriptive ability. The fitness score is very high (0.998), indicating that almost all the behaviors observed in the log are reproducible by the model. The precision is 0.645, suggesting that the model also accommodates behaviors not always observed in the log, an intended effect as the algorithm uses frequencies and causal dependencies to filter out the noise (making the model more flexible and realistic).
The high generalization (0.978) indicates a good ability to handle new future cases, an important characteristic in an e-commerce context where user behavior can vary over time (for example in cases such as promotions or new products). The simplicity value is very low (0.013), a symptom of a high structural complexity: the model is rich in nodes, arcs and dependencies, thus reflecting the variety of paths that the user can follow on the site but also being less immediate to interpret.

Therefore, the Heuristic Miner can be considered a very effective model for analyzing real and rumor processes such as those of an e-commerce, at the sole expense of visual complexity.

Inductive Miner



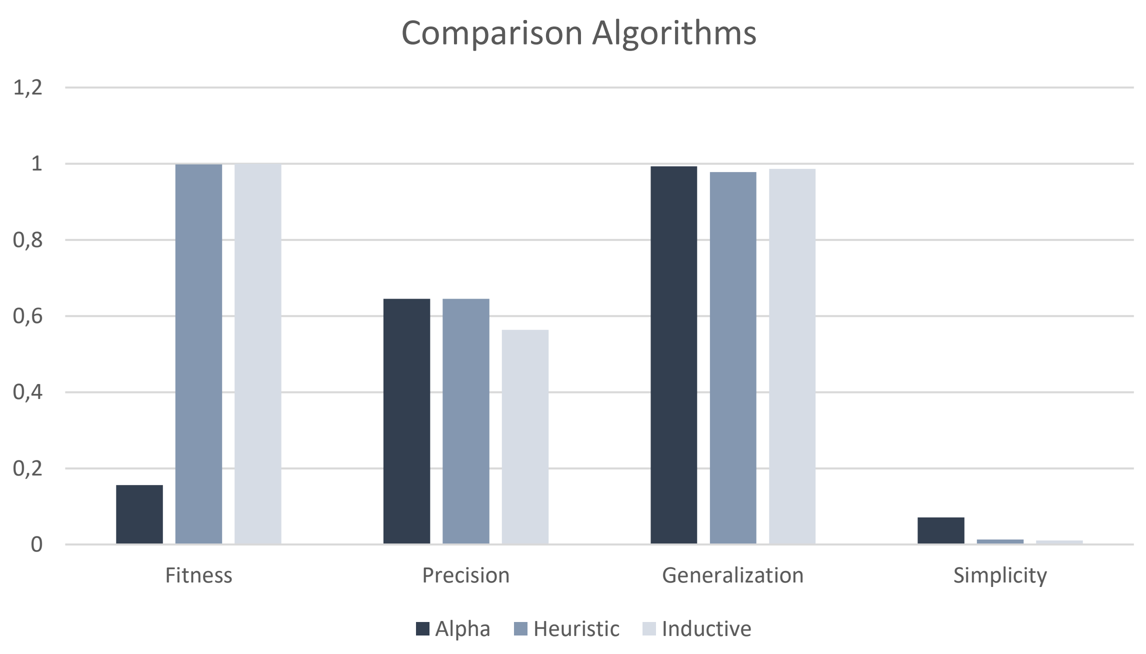
Inductive Miner Petri Net.

Fitness	Precision	Generalization	Simplicity
1.000	0.564	0.986	0.011

The Inductive Miner model achieves a maximum fitness score of (1.000), indicating that all cases in the event log can be reproduced without exception. This information is particularly important for an online purchasing process, where it is crucial not to miss any observed behavior.
However, the precision is lower than other models (0.564), indicating that the model also allows for paths not explicitly present in the log.
However, this characteristic is often acceptable for predictive analysis purposes.
The generalization is very high (0.986), confirming that the model is robust and suitable for handling future changes in user behavior, such as new navigation flows or changes to the website's web interface.

The Simplicity value of 0.011 indicates that the model generated by the Inductive Miner is structurally complex. This result is not a problem, on the contrary, it is consistent with the algorithm's goal, which is to prioritize the completeness and correctness of the model at the expense of its readability.

In conclusion, in a context such as that of a cosmetics e-commerce (characterized by high variability of user paths, repetitive and non-linear behaviors and presence of noise in the logs) the algorithms show different behaviors from each other: Alpha Miner is unsuitable for realistic analyses, Heuristic Miner instead offers a good compromise between fidelity and interpretability, while Inductive Miner emerges as a solid solution for complete and generalizable analyses, although sacrificing precision and simplicity.



3. Next Event Prediction

3.1 Description

A specific part of the project is dedicated to the probabilistic prediction function of what the next event will be, considering a sequence of transitions as input.
While traditional analytics only permit to understand what happened in the past, introducing a predictive layer permit to guide future decisions based on specific predictions.
With this approach, the event log is no longer considered a simple historical archive of interactions but becomes a decision-making tool to support the digital store in optimizing the user experience.
From this perspective, each user interaction is interpreted as part of a temporal trajectory, that is, an ordered sequence of actions that describes the evolution of behavior within the platform.
By analyzing the frequency and distribution of transitions between one activity and its next, the system builds a probabilistic transition matrix that describes the tendency of users to move from one state to another.

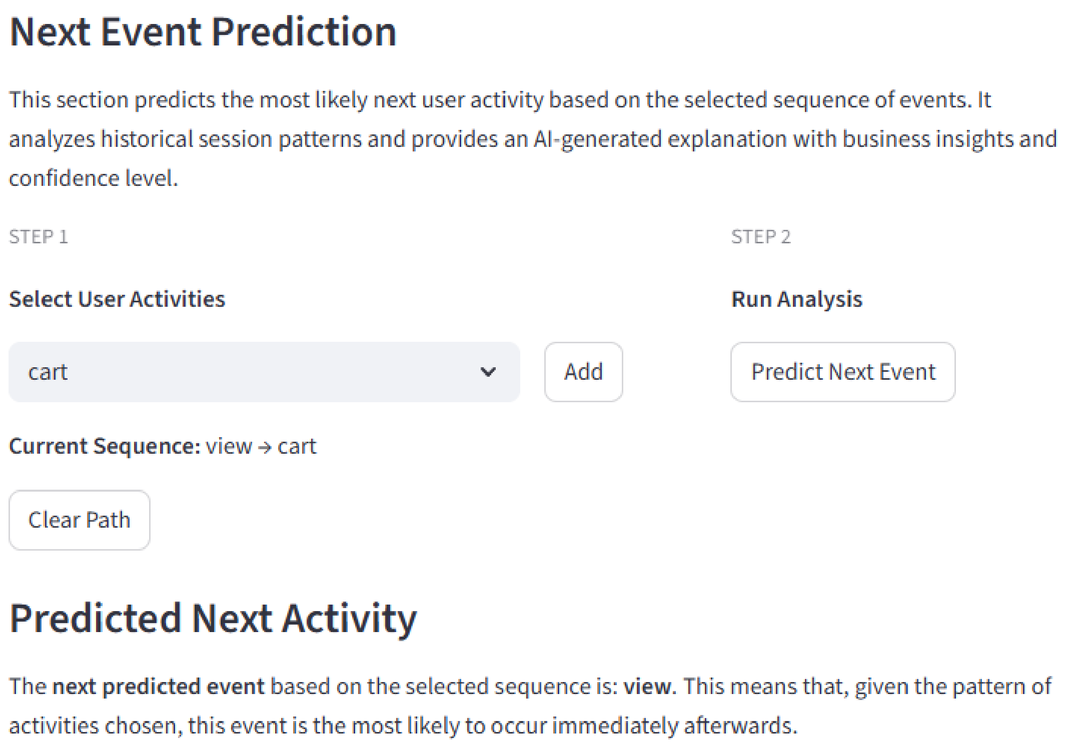
3.1 Applications

Predicting the next event is based on analyzing historical user behavior patterns. The system doesn't use machine learning or deep learning models, but simply uses frequency-sequential logic, observing what happened most often in the past when a specific sequence of actions occurred.

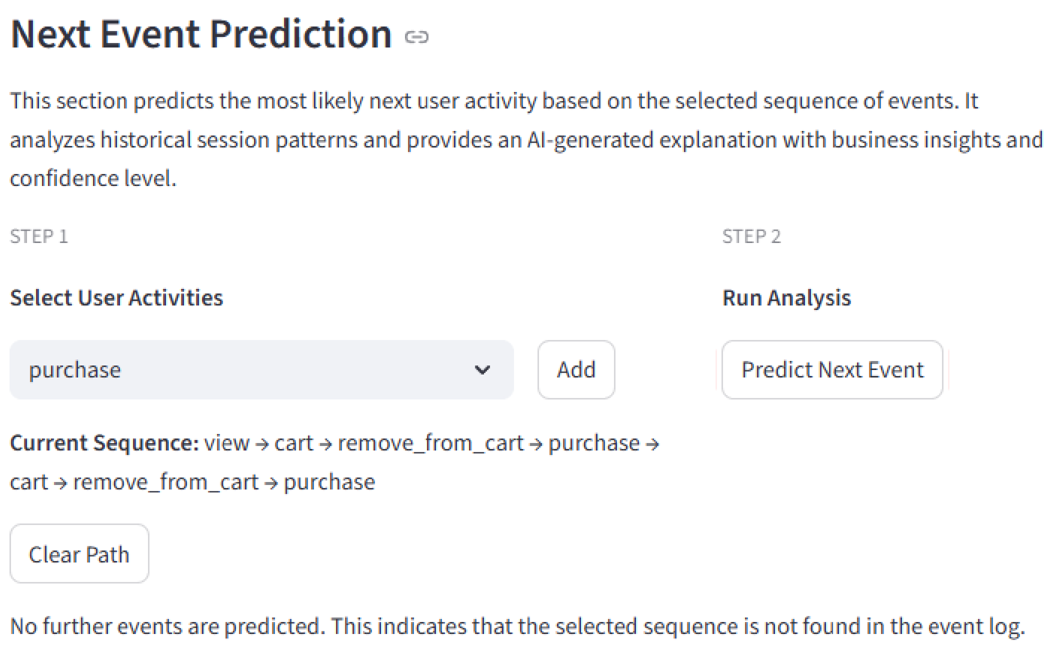
The starting point is the normalized event log: events are grouped by session and sorted chronologically (this already happens in pre-processing but is repeated for safety; providing an inaccurate chronological sequence would result in the generation of a prediction that is certainly incorrect). Traces are generated, i.e. real paths with all the history that occurs within a single session.
Once a specific sequence of events in a given order is input, the system scans all stored traces and checks how many times it occurs within the actual paths. Each time it occurs, the system checks and records which event occurs immediately after. This creates a statistical distribution of transitions, within which the event that occurs most often is considered the event statistically most likely to occur next.

If the entered sequence never appears in historical traces, the system does not have sufficient statistical basis to generate a prediction. In this scenario, a status equivalent to “None” will be returned, indicating that there is not enough information to infer a reliable continuation. This usually happens when the entered sequence is too specific (and therefore long) or when the datasets are too small to make predictions on.

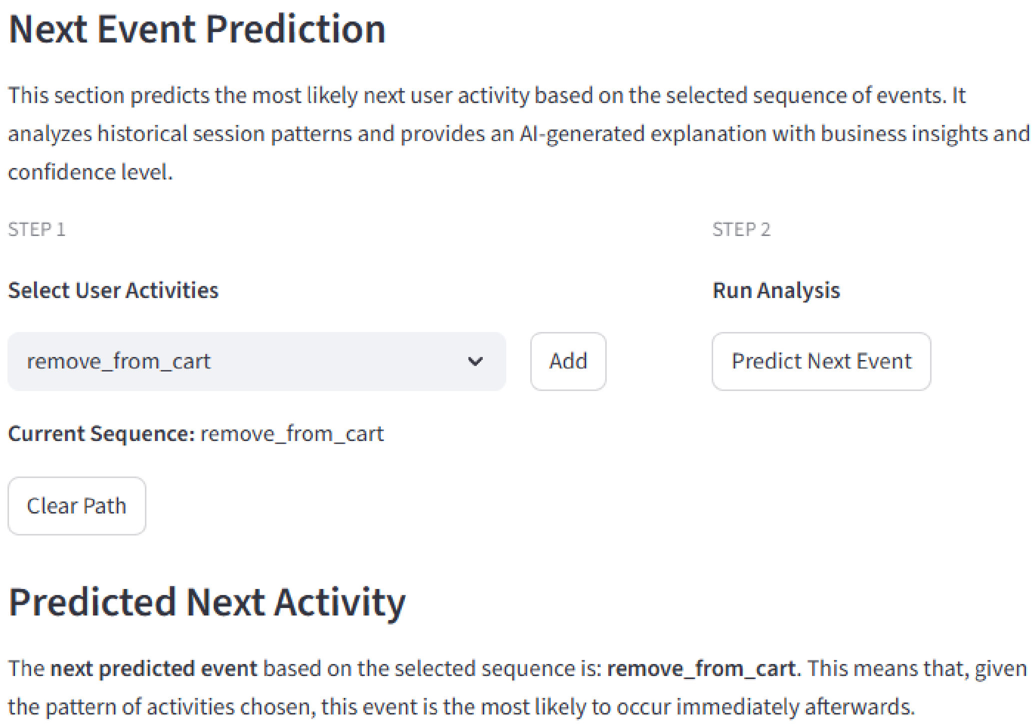
Below are reported some examples of input sequences and prediction of the next event.



Next Event Prediction, example n1.



Next Event Prediction, example n2.



Next Event Prediction, example n3.

4. LLM

4.1 Introduction

In the project, Artificial intelligence plays a very important and strategic role within several of the project's functionalities.
In particular, Large Language Models (LLM) are used to enrich data analysis with semantic interpretation capabilities, report generation and decision support.

From a technological perspective, LLM integration occurs through two main tools: the OpenAI Python library and the OpenRouter platform.
The OpenAI library allows direct access to next-generation language models via API, offering text generation, semantic analysis and contextual reasoning capabilities.
OpenRouter, on the other hand, is used as a routing and request management layer to LLM models, allowing for the use of a free API key and access to different models while maintaining a single communication interface.
This approach allows for greater architectural flexibility, as well as the ability to experiment with multiple solutions without being tied to a single provider.

The model initially selected for the project was "LLaMA 3.3 70B," one of the most advanced large-scale language models available free of charge on OpenRouter. This model is characterized by a high capacity for semantic understanding, contextual reasoning and structured text generation. During the operational phase of the project, however, using LLaMA 3.3 70B via OpenRouter revealed instability issues, resulting in frequent crashes and request interruptions. To ensure analysis continuity and reproducibility of results, the NVIDIA Nemotron-3 Nano 30B model, also available free of charge from OpenRouter, was chosen.
Despite being smaller than LLaMA 3.3 70B, Nemotron 3 Nano 30B offers a good compromise between performance, inference speed and stability.
Qualitatively, the Nemotron 3 Nano 30B demonstrated good natural language understanding, producing consistent and sufficiently accurate results.

Although its performances are inferior to a 70 billion parameter model in terms of reasoning depth, the increased reliability and reduced occurrence of crashes on the OpenRouter platform justify its temporary adoption for generating results.

Within the system, artificial intelligence is used in three macro-functional areas: generation of an interpretative report of the results of the prediction of the next event given as input a sequence of events, generation of a statistical report on the details of the specific dataset and the possibility of conversing with an artificial intelligence chatbot in relation to the results produced by the various process discovery algorithms.

4.2 Applications

The first application involves predicting the next event within a sequence of user activities. This section is directly linked to what was explained in the previous chapter and involves (after the statistical module identifies the most likely action based on historical frequencies) the intervention of the LLM with an interpretation and explanation function.
The AI essentially generates a textual report that explains the reasons for this prediction, providing an explanation that also addresses business concerns.
In addition to the explanation, the model also produces three to five operational recommendations aimed at improving process efficiency and optimizing the user experience.
At the end of the report, the system also provides an assessment of the prediction's confidence level.

AI Insight

This AI-generated insight explains the behavioral logic behind the selected sequence of activities, clarifies why the predicted next event is the most likely outcome, and lists actionable business recommendations to improve and optimize the process performance.

Reasoning
The observed user path is **['view', 'cart']**. In typical e-commerce process flows, after a user adds a product to the shopping cart, the most common subsequent actions are:

1. **Proceed to checkout** (purchase or checkout steps).
2. **Continue browsing** – i.e., view another product page, look at related items, or explore similar categories.
3. **Remove from cart / abandon** – less frequent unless the user encounters issues.

Empirical studies and process-mining analyses on cosmetics platforms consistently show that **view** is the dominant transition from *cart* when the user has not yet committed to purchase. This can be attributed to several behavioral patterns:

- **Exploratory intent**: Cosmetics shoppers often compare shades, finishes, or finishes across multiple items before deciding.
- **Decision-making latency**: The “add-to-cart” step is frequently a “save for later” gesture; users may still be evaluating alternatives.
- **Navigation loops**: After adding an item, the UI often surfaces “related products” or “you may also like,” prompting another **view** event.

Given that the only activities defined in the log are *view, cart, remove_from_cart,* and *purchase*, and that the system's transition matrix (derived from historical case data) shows the highest probability from *cart* to *view* (≈45 % of cart cases), the most likely next activity is another **view**.

Why not purchase or remove_from_cart?

- **Purchase** typically follows *checkout* or *cart* only after a *view* of the checkout page or a *cart*-to-checkout transition, which is not present in the current path.
- **Remove_from_cart** occurs when users encounter friction (e.g., price surprise, out-of-stock) and is observed in only ~10 % of cart cases.

Thus, based on frequency, contextual UI cues, and historical transition probabilities, **view** is the statistically most probable next step.

Business Actions

1. **Dynamic "Related Products" Recommendations**
    - Deploy a recommendation engine that surfaces complementary or alternative items immediately after a product is added to the cart.
    - Use real-time inventory and popularity data to suggest trending shades or complementary products, encouraging an additional **view** and potentially increasing basket size.
2. **Optimize Add-to-Cart Confirmation Flow**
    - Introduce a clear, frictionless "Continue Shopping" button that leads directly to a curated product-view page.
    - Reduce the number of clicks required to return to browsing, thereby reinforcing the natural **view → cart → view** loop and preventing premature checkout attempts that could cause abandonment.
3. **Personalized Cart-Retention Messaging**
    - Trigger targeted email or on-site notifications after a user leaves the cart (e.g., "You might also like…") that include a **view** link to related items.
    - Leverage past purchase history to tailor suggestions, increasing the likelihood of re-engagement and subsequent conversion.
4. **A/B Test Checkout Path Simplification**
    - Experiment with a streamlined checkout that can be accessed directly from the cart without leaving the browsing context.
    - Measure impact on transition probabilities; if checkout rates increase, adjust UI to guide users from **view** to checkout more efficiently.
5. **Monitor Drop-off Signals**
    - Set up process-mining dashboards to flag cases where *view* follows *cart* but no further activity occurs within a defined time window (e.g., 30 minutes).
    - Use these insights to identify potential friction points and iteratively improve product-detail page design or loading times.

Confidence
High – The prediction is grounded in a well-documented transition probability (cart → view) that consistently exceeds other possible successors in historical process logs. The reasoning incorporates both statistical frequency and contextual UI behavior, leaving little room for alternative high-probability outcomes.

The second application implements a structured procedure that, after loading the event log and converting it into a PM4Py-compatible format, calculates a series of quantitative indicators that form the information base of the model. These include: the total number of sessions, the average, minimum and maximum number of events per session, the most frequent activity, the most common exit activity and the number of unique process variants.
A summary of the main process variants is also added to these metrics, which describe the most recurring user paths and allow for the identification of dominant navigation patterns. The LLM therefore does not perform direct statistical calculations but receives a set of pre-processed information and transforms it into a discursive and strategic report. The LLM is required to interpret the numbers, identify potential bottlenecks and highlight user experience issues (proposing appropriate recommendations on how to proceed).
The final result is a textual document structured in 3 sections (Reasoning, Observations, Recommendations) that combines the visualization of simple metrics with strategic activities aimed at continuous improvement of the system.

AI Strategic Process Report

This AI strategic report is based on the overall data summary, including total user sessions, average session duration, and the most frequent process variants. It analyzes behavioral patterns, identifies potential bottlenecks, and provides actionable business recommendations with confidence levels.

Generate AI Strategic Report

All informations:

- DATA SUMMARY:
  - Total Sessions: 10,000
  - Average Events per Session: 11.82
  - Max Events in a Session: 585
  - Min Events in a Session: 2
  - Most Frequent Activity: **view**
  - Most Common Exit Activity: **view**
  - Unique Process Variants: **3,431**
- TOP PROCESS VARIANTS:
  1. `view → view` (2,019 occurrences)
  2. `view → view → view` (809 occurrences)
  3. `view → cart` (613 occurrences)
  4. `view → view → view → view` (426 occurrences)
  5. `view → view → view → view → view` (259 occurrences)

Reasoning

The process snapshot shows a **highly browsing-centric journey**: the dominant activity is "view," and the most frequent exit is also "view." The top variants are simple chains of repeated "view" events, with only a modest share (≈6 % of sessions) progressing to a "cart" step. The sheer number of unique variants (3,431) indicates that each user's path is fairly individualized, likely driven by personalized product displays, filters, or dynamic content.

Given the large number of sessions ending on a view and the relatively low conversion to cart, there appears to be **friction after product discovery**—users may be unable to find compelling reasons to add items to the cart, or the path to purchase is not clear enough.

Observations

- **Bottleneck at conversion:** The most common exit activity is "view," suggesting many users abandon the site without proceeding to cart or checkout.
- **Long view chains:** Repeated "view → view → …" sequences (up to five views) may indicate decision fatigue or lack of product relevance, leading to prolonged browsing without purchase.
- **High variant diversity:** 3,431 unique paths imply a fragmented user experience; personalization may be over-engineered, creating many isolated flows that are hard to optimize.
- **Low cart-entry frequency:** Only ~6 % of sessions contain a "view → cart" transition, highlighting a conversion gap.
- **Session length variance:** Sessions range from 2 to 585 events, showing both very short (likely bounce) and extremely long interactions, which can mask underlying usability issues.

Recommendations

#	Actionable Strategic Improvement	Confidence
1	**Introduce clearer product recommendation cues** (e.g., "Customers also bought," "Frequently viewed together") on the product detail page to guide users toward items they are likely to add to cart.	**High**
2	**Simplify navigation and reduce view fatigue** by limiting the number of "view" steps before a clear call-to-action (e.g., "Add to Cart" button highlighted after the first or second product view). Use progressive disclosure to surface key purchase information early.	**Medium**
3	**Implement a guided checkout funnel** with a single, prominent "Add to Cart → Checkout" pathway, possibly via a modal or sticky banner that appears after a user has viewed a product a set number of times (e.g., 2–3 views).	**High**
4	**Personalize the variant landscape** by clustering similar process paths and presenting a reduced set of curated journeys (e.g., "Popular path: view → view → cart"). This can decrease the total unique variants and make optimization easier.	**Medium**
5	**Run A/B tests on exit-intent offers** (e.g., discount pop-ups or free-shipping alerts) triggered when the exit activity is "view." Measure impact on conversion to cart and overall session length.	**Medium**

The third application is an artificial intelligence chatbot capable of answering questions about process discovery results in real time.
The chatbot has knowledge of the models generated by Alpha Miner, Heuristic Miner and Inductive Miner, including their respective quality metrics.
This component enables dynamic interaction with the analysis results: the user can ask questions about process flows, request clarification on the differences between algorithms, or request explanations of quality metrics.

Are there anomalies? Any types?

Yes – anomalies are present in all three discovered models.
They can be classified according to the metric patterns they produce.

Algorithm	Anomalous Metric Pattern	Type of Anomaly (as defined by the metric profile)
Alpha Miner	Fitness = 0.16 (very low) while Precision = 1.00 and Generalization = 0.99	**Coverage / Replay Anomaly** – the model fails to reproduce the majority of observed traces, indicating missing-behaviour or incomplete workflow.
Heuristic Miner	Precision = 0.65 (moderate) **and** Simplicity = 0.01 (near-zero)	**Over-parameterisation Anomaly** – the net contains many rarely used constructs, producing a highly complex model despite perfect fitness. The low simplicity signals structural noise.
Inductive Miner	Precision = 0.56 (low) **and** Simplicity = 0.01 (near-zero)	**Noise-Induced / Over-fitting Anomaly** – the discovered net fits the log perfectly (Fitness = 1.00) but does so by introducing many spurious paths, reflected by low precision and negligible simplicity.

Summary of anomaly types

1. **Coverage/Replay Anomaly** – Alpha Miner's fitness far below 1.0, meaning a large portion of the event-log behavior cannot be replayed.
2. **Over-parameterisation Anomaly** – Heuristic Miner's precision is acceptable but its simplicity score is essentially zero, indicating an unnecessarily large, tangled net.
3. **Noise-Induced / Over-fitting Anomaly** – Inductive Miner also achieves perfect fitness but does so with a very low-precision, low-simplicity net, suggesting the model captures log noise rather than meaningful process structure.

These anomalies highlight that while Heuristic and Inductive miners satisfy the "fitness = 1" requirement, their low precision and simplicity scores reveal structural problems that must be addressed (e.g., pruning rarely used branches, tightening relation-frequency thresholds) before the models can be considered operationally useful.

Can you tell me about your optimization strategies?

Optimization levers that can be evaluated with the provided metrics

Goal	Metric(s) to watch	Typical adjustment (within the three supported algorithms)	Expected metric impact
Increase precision without sacrificing fitness	Precision (target > 0.65)	– Prune rarely-used branches that contribute little to trace coverage. – Raise the frequency threshold for including relations in the Heuristic/Inductive net.	Precision rises; Fitness usually stays at 1.00 because the retained structure still reproduces all traces.
Improve simplicity while keeping fitness	Simplicity (target > 0.01)	– Apply a "noise-filter" step that removes self-loops or duplicate edges that never appear in the log. – Collapse parallel moves that have identical pre- and post-conditions.	Simplicity improves (score moves toward 1.00); Precision may dip slightly if too aggressive, so monitor the trade-off.
Maintain high generalization	Generalization (target ≈ 0.99)	– Limit the depth of the discovery tree (e.g., stop splitting when a relation frequency falls below a configurable cut-off). – Use a more restrictive relation-frequency filter before constructing the net.	Generalization remains high; may help keep Precision stable when pruning.
Recover from low fitness (as seen with Alpha Miner)	Fitness (target > 0.16)	– Switch to a more expressive algorithm (Heuristic or Inductive) because Alpha Miner's sequential-only assumption cannot capture parallelism. – Enrich the event log with additional attributes (e.g., case attributes) that enable richer relation discovery, then re-apply Heuristic/Inductive.	Fitness can jump to 1.00; Precision and Simplicity will be governed by the new algorithm's settings.

Key take-aways for optimization

1. **Prioritize fitness first** – a model that cannot replay the majority of traces (Fitness = 0.16) is unusable, regardless of other scores.
2. **Balance precision and simplicity** – after guaranteeing perfect fitness, focus on raising Precision (to reduce spurious behavior) and then on lifting Simplicity (to obtain a more compact, maintainable model).
3. **Use metric-driven thresholds** – for example, only retain relations that appear in ≥ 5 % of traces; this typically boosts Precision and Simplicity while leaving Fitness untouched.
4. **Iteratively evaluate** – after each structural change, recompute the four metrics. If Precision drops sharply, relax the pruning criteria; if Simplicity remains near zero, consider merging equivalent activities or collapsing duplicate edges.

These adjustments stay within the confines of the three supported discovery algorithms and the four quality metrics you are tracking, allowing you to systematically improve the most useful model (the Heuristic Miner) for your use case.

Fine, which of the three models is the most useful for our use case? Why?

Most useful model: Heuristic Miner

Why:

Metric	Heuristic Miner	Inductive Miner	Alpha Miner
Fitness	1.00 (perfect trace coverage)	1.00 (perfect)	0.16 (covers only ≈ 16 % of observed behavior)
Precision	0.65 (moderate; the highest among the three that achieve perfect fitness)	0.56 (lower)	1.00 (high, but irrelevant because fitness is poor)
Generalization	0.98 (very good)	0.99 (slightly better, but marginal)	0.99 (very good)
Simplicity	0.01 (low, comparable to Inductive)	0.01 (low)	0.07 (higher, but offset by poor fitness)

- **Fitness is the decisive factor** for practical usefulness: a model must be able to replay the majority of observed cases. Both Heuristic and Inductive achieve perfect fitness, whereas Alpha fails dramatically.
- **Precision** distinguishes the two perfect-fitness models. Heuristic's 0.65 is substantially higher than Inductive's 0.56, indicating fewer spurious or rarely used constructs.
- **Generalization** is comparable for Heuristic (0.98) and Inductive (0.99); the tiny difference does not outweigh the precision advantage.
- **Simplicity** is equally low for Heuristic and Inductive (0.01), so it does not provide a differentiator. The low simplicity is a known limitation of both, but it does not outweigh the operational benefit of higher precision.

Conclusion:
Given the metric-driven evaluation, the Heuristic Miner offers the best balance—perfect trace coverage *and* the highest precision among the viable options—making it the most useful for the described use case. The Inductive Miner, while equally fit, introduces more unnecessary complexity, and the Alpha Miner is unsuitable because it cannot reproduce the majority of process behavior.

5. Observations for future improvement

The project, through the development of an interactive and functional dashboard based on Streamlit, achieved its objectives, enabling the processing and elaboration of a CSV dataset of event logs from a cosmetics e-commerce site to generate not only models using algorithms such as Alpha Miner, Heuristic Miner and Inductive Miner, but also predictive analysis and accurate reports with the support of artificial intelligence. The use of LLM was also particularly instrumental in identifying problems and anomalies, optimizing processes and identifying bottlenecks and anomalies.

Based on the case study in question, some observations could be made for possible future improvements of the application.

- The AI-based chatbot could have deeper knowledge. Currently, the AI chatbot's knowledge is limited to analyzing Process Discovery algorithms to answer questions related to this topic in real time. It can effectively answer questions that involve analyzing Process Mining quality metrics, compare the performance of metrics-based algorithms, identifying weaknesses, trade-offs, or inconsistencies in the discovered models, detecting potential anomalies or overfitting/underfitting patterns and providing actionable insights based solely on the observed metrics. This is sufficient to effectively address the case study's design requirements, but it could be useful to introduce new knowledge in the future. The chatbot could also be aware of the information generated by the strategic report, such as the number of unique sessions, their average path, the shortest path, the longest path and much more. Expanding the chatbot would be nothing more than an expansion of its knowledge and functionality, making it a true 360-degree support tool for every use case.
- The system prompts and user prompts could be improved and refined to generate even more accurate and specific results. This would be achieved following a lengthy analysis and testing phase, during which various LLMs and prompts would be tested to determine the best solutions for our specific use case.

  The results currently generated by the models are already specific and adequately meet the case study's requirements. However, the use of paid models (with a broader context) and the availability of unlimited daily calls to the models would allow for a more thorough testing phase of the prompts and their results, ensuring even more effective reporting and chatbot responses.

- Predictions for the next probable event could be generated entirely by artificial intelligence. Currently, predictions are based on a simple static probabilistic analysis, based on a mathematical calculation of the next events in recorded sessions once the same sequence of events was presented. This statistical information could be fed to an artificial intelligence model and have it generate the prediction (and the subsequent associated report, as is already the case). This would improve the accuracy of the prediction, as the artificial intelligence, using specific deep learning algorithms and having even greater contextual knowledge at its disposal, would be able to make more realistic predictions.
- The size of the event logs considered could be increased, ensuring greater representativeness and greater model stability. This would increase the possibility of capturing rare process variants, improving the estimation of path frequencies or times. At the same time, the model would also be less sensitive to outliers or random noise, allowing frequent patterns to emerge more clearly and ensuring better stability.
  However, computational costs (computing time and memory) would increase exponentially, and the use of algorithms such as Alpha, Inductive, and Heuristic Miner would impact the model's complexity. For our use case, using 10.000 unique sessions is a good compromise for the objectives of the case study, but by varying the objectives, it might also be appropriate to increase the size of the events log.
- Perform continuous updates to process mining processes to obtain up-to-date and actionable data. If the system were continuously running on a server, continuous updates could be performed once a month, generating consistently up-to-date dashboard results.
  This should be done with a view to actual use at the corporate level, rather than a case study, always taking into account the latest events log in terms of time. This would ensure consistently current and up-to-date results regarding anomalies, predictions, or reports, ensuring a high level of operational efficiency at the production level.