Computer Science Department

Master Degree in Computer Science

Formal Methods in Computer Science

# Using Process Mining and LLMs for Activity Recognition in a Smart Home Environment

Author:
**Mattia Curri**

ACADEMIC YEAR **2024-2025**

# Contents

# 1 | Introduction

## Process Mining

Process mining is a set of techniques and tools aimed at analyzing event logs to extract insights into how business or operational processes are actually carried out. It blends model-based process analysis and data mining techniques to provide an evidence-based approach to process discovery, conformance checking, and performance analysis.

Process mining can be divided into three main types:

- **Process Discovery**: This involves creating a process model from scratch based on event logs.
- **Conformance Checking**: This compares an existing process model with an event log of the same process to check if the real process conforms to the model.
- **Enhancement**: This involves improving an existing process model using information from event logs.

### Applications

Process mining techniques are widely used in industries such as healthcare, logistics, and finance to:

- **Identify Bottlenecks and Inefficiencies**: By analyzing event logs, organizations can identify where delays and inefficiencies occur in their processes.
- **Ensure Compliance with Established Procedures**: Process mining can be used to check if actual processes conform to regulatory requirements and internal policies.
- **Evaluate the Performance of Complex Workflows**: Organizations can assess the performance of their workflows and identify areas for improvement.
- **Provide Real-Time or Near Real-Time Monitoring of Ongoing Processes**: Process mining tools can provide real-time insights into ongoing processes, allowing for timely interventions.

## Main Algorithms

### Alpha Algorithm

The Alpha algorithm constructs a Petri net representation of a process by analyzing event logs. Here is a detailed step-by-step description of how the Alpha algorithm works:

1. **Identify Activities**: Extract all unique activities from the event log.
2. **Determine Direct Succession**: Identify pairs of activities where one directly follows the other.
3. **Identify Causality**: Determine causal relationships between activities based on direct succession.
4. **Identify Parallelism**: Identify activities that can occur concurrently.
5. **Construct Places**: Create places in the Petri net to represent the identified causal and parallel relationships.
6. **Connect Transitions and Places**: Connect activities (transitions) to places based on the identified relationships.
7. **Create Initial and Final Places**: Define the initial and final places in the Petri net to represent the start and end of the process.

### Heuristic Miner

The Heuristic Miner algorithm is an extension of the Alpha algorithm that deals with noise and incompleteness in event logs. It uses frequency-based heuristics to construct a more robust process model. Key steps include:

1. **Frequency Analysis**: Analyze the frequency of activity sequences in the event log.
2. **Threshold Setting**: Set thresholds to filter out infrequent and potentially noisy sequences.
3. **Construct Dependency Graph**: Create a dependency graph based on the filtered sequences.
4. **Generate Process Model**: Convert the dependency graph into a process model, typically a Petri net or a causal net.

### Inductive Miner

The Inductive Miner algorithm aims to produce sound process models that are guaranteed to be free of deadlocks and other anomalies. It works as follows:

1. **Log Splitting**: Recursively split the event log based on directly-follows relations.
2. **Base Case Handling**: Handle base cases where the log cannot be split further.

3. **Model Construction**: Construct process models for each split and combine them to form the final model.

4. **Guarantee Soundness**: Ensure that the resulting process model is sound and free of deadlocks.

#### Noise Threshold Parameter

The noise threshold parameter is used to filter out infrequent and potentially noisy behavior from the event logs. This parameter helps in creating a more accurate and robust process model by ignoring rare events that may not represent the typical process flow. The steps to apply the noise threshold parameter are:

1. **Set Threshold**: Define a threshold value that determines the minimum frequency for an event or sequence to be considered in the process model.

2. **Filter Events**: Remove events and sequences from the event log that occur less frequently than the defined threshold.

3. **Reconstruct Model**: Use the filtered event log to reconstruct the process model, ensuring that only significant and frequent behaviors are included.

4. **Validate Model**: Validate the resulting process model to ensure it accurately represents the main process flow without being affected by noise.

### Activity Recognition in Smart Homes

A relevant application involves human activity recognition in a smart home environment. By interpreting sensor data as event logs, process mining can be leveraged to discover patterns in daily human routines, detect anomalies, and improve assistive solutions. This helps researchers and practitioners identify how residents navigate spaces, complete tasks, and interact with devices, providing opportunities for intelligent interventions, personalization, and automation.

In a smart home environment, various sensors (e.g., motion sensors, door sensors, appliance usage sensors) generate event logs that capture the activities of residents. Process mining techniques can be applied to these logs to:

- **Discover Activity Patterns**: Identify common sequences of activities and routines.

- **Detect Anomalies**: Recognize deviations from typical patterns that may indicate issues such as health problems or security breaches.

- **Enhance Assistive Technologies**: Improve the performance of assistive technologies by providing insights into user behavior and preferences.

- **Enable Automation**: Automate routine tasks based on identified patterns, enhancing the convenience and efficiency of the smart home.

# 2 | Dataset

## Dataset Description

The dataset used in this study is provided in two formats: CSV and XES. It contains event logs from a smart home environment, capturing various activities performed by residents. The dataset is instrumental in analyzing human activity patterns and applying process mining techniques for activity recognition.

### CSV Format

The CSV file, named `mit_formatted_noproc_notrimmed.csv`, contains the following columns:

- **Timestamp**: The date and time when the event occurred.
- **CaseID**: The identifier for the specific case or instance of the process.
- **Activity**: The activity performed, including the device or sensor involved and the action taken.

An example of a row in the CSV file is as follows:

```
2003-03-27 06:43:40, Cabinet_67, Cabinet_67 ON Toileting
```

### XES Format

The XES file, named `mit_formatted_noproc_notrimmed.xes`, follows the eXtensible Event Stream (XES) standard for representing event logs. It includes detailed information about each event, such as:

- **Trace**: Represents a sequence of events corresponding to a specific case or instance of the process.
- **Event**: Contains attributes such as the timestamp, activity name, and other relevant details.

An example of an event in the XES file is as follows:

```
<event>
  <date key="time:timestamp" value="2003-03-27T06:43:40" />
  <string key="concept:name" value="Cabinet_67 ON Toileting" />
```

```
</event>
```

## Data Collection

The data was collected using various sensors installed in a smart home environment. These sensors include motion sensors, door sensors, appliance usage sensors, and more. Each sensor generates event logs that capture the activities of the residents, providing a comprehensive view of their daily routines.

## Applications

The dataset can be used for various applications, including:

- **Activity Recognition**: Identifying and classifying different activities performed by residents.
- **Anomaly Detection**: Detecting deviations from typical activity patterns that may indicate issues such as health problems or security breaches.
- **Assistive Technologies**: Enhancing the performance of assistive technologies by providing insights into user behavior and preferences.
- **Automation**: Automating routine tasks based on identified patterns, improving the convenience and efficiency of the smart home.

# 3 | Methodology

## Preprocessing

The preprocessing phase involved several steps to prepare the dataset for process mining analysis. The main steps included:

### CSV Formatting

The raw CSV file was formatted to ensure consistency and readability. The `format_csv` function was used to read the input CSV file and format each line by combining the date and time into a single timestamp, extracting the case ID, and concatenating the activity details.

### Removing Suffix Numbers

To generalize activity labels, the `remove_suf_num` function was applied. This function removed numerical suffixes from the `CaseID` and `Activity` columns, and grouped similar activities (e.g., combining `Sink_faucet_-_hot` and `Sink_faucet_-_cold` into `Sink_faucet`).

### Processing Activity Windows

The `process_csv` function was used to segment the activities into windows based on changes in activity or case ID. This function ensured that each activity window had a start and end timestamp, providing a clear view of the activity duration.

### Conversion to XES

The formatted CSV files were converted to XES format using the `csv_to_xes` function. This function read the CSV file, converted it to an event log, and saved it as an XES file. The conversion was performed for both trimmed and untrimmed versions of the dataset.

## Experiments

Several experiments were conducted to evaluate the performance of different process discovery algorithms on the preprocessed dataset. The experiments included:

### Base Model Experiments

- **No Processing, Base Model, No Trim**: The raw dataset was formatted and converted to XES without any additional processing or trimming. The Alpha Miner, Heuristic Miner, and Inductive Miner algorithms were applied, and their metrics were evaluated.

- **No Processing, Tuned Model, No Trim**: The raw dataset was formatted and converted to XES without any additional processing or trimming. The Alpha Miner and Heuristic Miner algorithms were applied with tuned parameters, and the Inductive Miner was evaluated with different noise thresholds.

### Processed Model Experiments

- **Processing, Base Model, No Trim**: The dataset was processed to remove suffix numbers and segment activity windows. The processed dataset was converted to XES without trimming. The Alpha Miner, Heuristic Miner, and Inductive Miner algorithms were applied, and their metrics were evaluated.

- **Processing, Tuned Model, No Trim**: The processed dataset was converted to XES without trimming. The Alpha Miner and Heuristic Miner algorithms were applied with tuned parameters, and the Inductive Miner was evaluated with different noise thresholds.

### Trimmed Model Experiments

- **No Processing, Tuned Model, Trim**: The raw dataset was formatted and converted to XES with trimming. The Alpha Miner and Heuristic Miner algorithms were applied with tuned parameters, and the Inductive Miner was evaluated with different noise thresholds.

- **Processing, Tuned Model, Trim**: The dataset was processed to remove suffix numbers and segment activity windows. The processed dataset was converted to XES with trimming. The Alpha Miner and Heuristic Miner algorithms were applied with tuned parameters, and the Inductive Miner was evaluated with different noise thresholds.

- **No Processing, Base Model, Trim**: The raw dataset was formatted and converted to XES with trimming. The Alpha Miner, Heuristic Miner, and Inductive Miner algorithms were applied, and their metrics were evaluated.

- **Processing, Base Model, Trim**: The dataset was processed to remove suffix numbers and segment activity windows. The processed dataset was converted to XES with trimming. The Alpha Miner, Heuristic Miner, and Inductive Miner algorithms were applied, and their metrics were evaluated.

### LLM Augmented Experiments

Experiments were also conducted using datasets augmented with data generated by large language models (LLMs). The augmented datasets were combined with the original dataset and converted to XES format. The Alpha Miner, Heuristic Miner, and Inductive Miner algorithms were applied, and their metrics were evaluated for different noise thresholds. All nets produced with LLMs dataset were compared against the original dataset and the augmented one.

## Metrics Evaluation

The performance of the process discovery algorithms was evaluated using the following metrics:

- **Fitness**: Measures how well the discovered model can reproduce the behavior observed in the event log.
- **Precision**: Measures how much behavior allowed by the model is observed in the event log.
- **Simplicity**: Measures the complexity of the discovered model.
- **Generalization**: Measures how well the model can generalize to unseen behavior.

The metrics were plotted for each experiment to compare the performance of the different algorithms and configurations.

# 4 | Results

In this section we will report the main results obtained from the experiments conducted in the previous chapter. We will analyze the performance of the process discovery algorithms on the pre-processed dataset and evaluate their fitness, precision, simplicity, and generalization. The results will provide insights into the effectiveness of different algorithms and configurations in discovering process models from event logs.

## 4.1 Base Model Experiments

| Miner Configuration | Fitness | Precision | Simplicity | Generalization |
|---|---|---|---|---|
| Inductive Miner (0.0) | **1.000** | 0.276 | 0.640 | 0.304 |
| Inductive Miner (0.1) | 0.999 | 0.356 | 0.643 | **0.308** |
| Inductive Miner (0.2) | 0.994 | 0.445 | 0.645 | 0.300 |
| Inductive Miner (0.3) | 0.989 | 0.511 | 0.648 | 0.294 |
| Inductive Miner (0.5) | 0.978 | 0.636 | 0.660 | 0.282 |
| Inductive Miner (0.6) | 0.976 | 0.629 | 0.661 | 0.278 |
| Inductive Miner (0.8) | 0.970 | 0.624 | 0.664 | 0.265 |
| Heuristic Miner | 0.955 | **0.698** | 0.588 | 0.232 |
| Alpha Miner | 0.554 | 0.025 | **1.000** | 0.297 |

Table 4.1: Performance metrics comparison. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

As we can see, the best model is the Heuristic Miner, with a precision of 0.698, while the Inductive Miner became better increasing the noise threshold, meaning that this dataset could be very noisy.

## 4.2 Processed Model Experiments

| Miner Configuration | Fitness | Precision | Simplicity | Generalization |
|---|---|---|---|---|
| Inductive Miner (0.0) | **0.999** | 0.111 | 0.633 | 0.407 |
| Inductive Miner (0.1) | **0.999** | 0.222 | 0.634 | 0.410 |
| Inductive Miner (0.2) | 0.991 | 0.176 | 0.636 | 0.407 |
| Inductive Miner (0.3) | 0.985 | 0.337 | 0.640 | 0.405 |
| Inductive Miner (0.5) | 0.969 | 0.389 | 0.647 | 0.344 |
| Inductive Miner (0.6) | 0.969 | 0.368 | 0.646 | 0.349 |
| Inductive Miner (0.8) | 0.975 | 0.382 | 0.650 | 0.318 |
| Heuristic Miner | 0.973 | 0.450 | 0.558 | 0.294 |
| Alpha Miner | 0.425 | 0.015 | 1.000 | 0.412 |

Table 4.2: Performance metrics comparison. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

As we can see, the processing doesn't improve the performance, it worsens it.

## 4.3 Trimmed Dataset Experiments

| Miner Configuration | Fitness | Precision | Simplicity | Generalization |
|---|---|---|---|---|
| Inductive Miner (0.0) | **1.000** | 0.262 | 0.641 | 0.306 |
| Inductive Miner (0.1) | 0.999 | 0.338 | 0.644 | **0.310** |
| Inductive Miner (0.2) | 0.994 | 0.424 | 0.646 | 0.302 |
| Inductive Miner (0.3) | 0.988 | 0.489 | 0.649 | 0.297 |
| Inductive Miner (0.5) | 0.976 | 0.617 | 0.662 | 0.284 |
| Inductive Miner (0.6) | 0.974 | 0.610 | 0.663 | 0.279 |
| Inductive Miner (0.8) | 0.968 | 0.605 | 0.665 | 0.266 |
| Heuristic Miner | 0.982 | **0.677** | 0.588 | 0.233 |
| Alpha Miner | 0.533 | 0.022 | **1.000** | 0.299 |

Table 4.3: Performance metrics comparison. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners. Best results per column are highlighted.

As we can see, the performance are not so different from the no trimmed dataset.

## 4.4 LLM Augmented Experiments

In this section we will compare the performances against the original dataset and the augmented one.

### 4.4.1 System Prompt

Here there is the system prompt provided. Note that only GPT allows to use it as system prompt in the strict sense, while the other LLMs received this as first message of a new conversation:

System Prompt

```
You are a synthetic data generator specialized in process
mining. Your task is to expand the user-provided dataset
(with columns [Timestamp, CaseID, Activity]) by creating
new realistic cases that strictly adhere to the existing
activities in the original dataset.
Binding Instructions:
Mandatory Input: The user will always provide an existing
dataset as reference.
Activity Preservation: Use only the activities present in
the original dataset (no new invented activities).
Output Structure: Maintain the exact 3-column order:
Timestamp, CaseID, Activity.

Generation Guidelines:
Temporal Patterns:
Create plausible chronological sequences based on observed
logic in the original data.

CaseID Generation:
CaseIDs represent sensors from which the data was
obtained. Do not invent new CaseIDs.
Follow the original CaseID format (e.g., C_1001, S_205).

Realistic Timestamps:
Ensure intra-case chronological consistency (no time
travel within a case).

Required Output Example:

Timestamp,CaseID,Activity
2023-11-05T08:02:34Z,C_1001,Start
2023-11-05T08:03:11Z,C_1001,Material_Loading
2023-11-05T08:07:22Z,C_1001,Quality_Check
2023-11-05T08:09:45Z,C_1001,End

Absolute Constraints:
Do not invent new activities or CaseIDs.
Do not alter column order or add/remove fields.

Use Cases:
Enrich training data for process discovery algorithms.
```

### 4.4.2 Provided Data

Given the limited context window, we choose to provide the LLMs data about the less frequent activities in the original dataset, in order to see if they can generate new cases with those activities to obtain better results.

### 4.4.3 Results

| GPT-4o (Original Dataset) | | | | |
|---|---|---|---|---|
| **Miner Configuration** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.271 | 0.640 | **0.299** |
| Inductive Miner (0.2) | 0.994 | 0.428 | 0.644 | 0.293 |
| Inductive Miner (0.5) | 0.978 | 0.599 | 0.659 | 0.273 |
| Inductive Miner (0.6) | 0.976 | 0.593 | 0.660 | 0.269 |
| Inductive Miner (0.8) | 0.970 | 0.589 | 0.662 | 0.256 |
| Heuristic Miner | 0.955 | **0.664** | 0.589 | 0.227 |
| Alpha Miner | 0.554 | 0.025 | **1.000** | 0.291 |

Table 4.4: Performance metrics comparison **on the original dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| GPT 4o (Augmented Dataset) | | | | |
|---|---|---|---|---|
| **Miner** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.298 | 0.640 | **0.305** |
| Inductive Miner (0.2) | 0.995 | 0.474 | 0.644 | 0.301 |
| Inductive Miner (0.5) | 0.979 | 0.657 | 0.659 | 0.278 |
| Inductive Miner (0.6) | 0.978 | 0.656 | 0.660 | 0.278 |
| Inductive Miner (0.8) | 0.973 | 0.650 | 0.662 | 0.267 |
| Heuristic Miner | 0.947 | **0.730** | 0.589 | 0.234 |
| Alpha Miner | 0.533 | 0.029 | **1.000** | 0.299 |

Table 4.5: Performance metrics comparison **on the augmented dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Claude 3.5 Sonnet (Original Dataset) | | | | |
|---|---|---|---|---|
| **Miner Configuration** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **0.987** | 0.261 | 0.635 | 0.279 |
| Inductive Miner (0.2) | 0.980 | 0.413 | 0.645 | 0.273 |
| Inductive Miner (0.5) | 0.967 | 0.563 | 0.650 | 0.257 |
| Inductive Miner (0.6) | 0.966 | 0.567 | 0.652 | 0.258 |
| Inductive Miner (0.8) | 0.955 | 0.577 | 0.654 | 0.246 |
| Heuristic Miner | 0.958 | **0.694** | 0.589 | 0.212 |
| Alpha Miner | 0.589 | 0.024 | **1.000** | **0.291** |

Table 4.6: Performance metrics comparison **on the original dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Claude 3.5 Sonnet (Augmented Dataset) | | | | |
|---|---|---|---|---|
| **Miner Configuration** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.266 | 0.635 | 0.302 |
| Inductive Miner (0.2) | 0.990 | 0.425 | 0.645 | 0.297 |
| Inductive Miner (0.5) | 0.972 | 0.579 | 0.650 | 0.282 |
| Inductive Miner (0.6) | 0.972 | 0.583 | 0.652 | 0.281 |
| Inductive Miner (0.8) | 0.962 | 0.593 | 0.654 | 0.273 |
| Heuristic Miner | 0.951 | **0.697** | 0.589 | 0.231 |
| Alpha Miner | 0.536 | 0.025 | **1.000** | **0.325** |

Table 4.7: Performance metrics comparison **on the augmented dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Qwen 2.5 Plus (Original Dataset) | | | | |
|---|---|---|---|---|
| **Miner** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.239 | 0.639 | **0.281** |
| Inductive Miner (0.2) | 0.994 | 0.338 | 0.643 | 0.270 |
| Inductive Miner (0.5) | 0.978 | 0.419 | 0.655 | 0.246 |
| Inductive Miner (0.6) | 0.976 | 0.416 | 0.656 | 0.243 |
| Inductive Miner (0.8) | 0.970 | 0.416 | 0.658 | 0.230 |
| Heuristic Miner | 0.955 | **0.535** | 0.597 | 0.213 |
| Alpha Miner | 0.554 | 0.025 | **1.000** | 0.237 |

Table 4.8: Performance metrics comparison **on the original dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Qwen 2.5 Plus (Augmented Dataset) | | | | |
|---|---|---|---|---|
| **Miner** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.434 | 0.639 | **0.281** |
| Inductive Miner (0.2) | 0.997 | 0.621 | 0.643 | 0.270 |
| Inductive Miner (0.5) | 0.988 | 0.772 | 0.655 | 0.242 |
| Inductive Miner (0.6) | 0.987 | 0.770 | 0.657 | 0.240 |
| Inductive Miner (0.8) | 0.984 | 0.764 | 0.658 | 0.229 |
| Heuristic Miner | 0.800 | **0.852** | 0.597 | 0.213 |
| Alpha Miner | 0.754 | 0.064 | **1.000** | 0.237 |

Table 4.9: Performance metrics comparison **on the augmented dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Deepseek-R1 (Original Dataset) | | | | |
|---|---|---|---|---|
| **Miner** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.245 | 0.639 | **0.276** |
| Inductive Miner (0.2) | 0.994 | 0.353 | 0.642 | 0.265 |
| Inductive Miner (0.5) | 0.978 | 0.447 | 0.654 | 0.240 |
| Inductive Miner (0.6) | 0.976 | 0.444 | 0.655 | 0.237 |
| Inductive Miner (0.8) | 0.970 | 0.443 | 0.657 | 0.224 |
| Heuristic Miner | 0.956 | **0.509** | 0.592 | 0.206 |
| Alpha Miner | 0.554 | 0.024 | **1.000** | 0.246 |

Table 4.10: Performance metrics comparison **on the original dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Deepseek-R1 (Augmented Dataset) | | | | |
|---|---|---|---|---|
| **Miner** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.408 | 0.639 | **0.282** |
| Inductive Miner (0.2) | 0.997 | 0.596 | 0.642 | 0.272 |
| Inductive Miner (0.5) | 0.986 | 0.754 | 0.654 | 0.245 |
| Inductive Miner (0.6) | 0.985 | 0.753 | 0.655 | 0.245 |
| Inductive Miner (0.8) | 0.982 | 0.747 | 0.657 | 0.234 |
| Heuristic Miner | 0.940 | **0.850** | 0.592 | 0.213 |
| Alpha Miner | 0.690 | 0.052 | **1.000** | 0.259 |

Table 4.11: Performance metrics comparison **on the augmented dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Mistral (Original Dataset) | | | | |
|---|---|---|---|---|
| **Miner** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.258 | 0.640 | **0.282** |
| Inductive Miner (0.2) | 0.994 | 0.388 | 0.644 | 0.272 |
| Inductive Miner (0.5) | 0.978 | 0.514 | 0.657 | 0.247 |
| Inductive Miner (0.6) | 0.976 | 0.510 | 0.658 | 0.244 |
| Inductive Miner (0.8) | 0.970 | 0.508 | 0.660 | 0.232 |
| Heuristic Miner | 0.955 | **0.593** | 0.591 | 0.215 |
| Alpha Miner | 0.554 | 0.025 | **1.000** | 0.274 |

Table 4.12: Performance metrics comparison **on the original dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

| Mistral (Augmented Dataset) | | | | |
|---|---|---|---|---|
| **Miner** | **Fitness** | **Precision** | **Simplicity** | **Generalization** |
| Inductive Miner (0.0) | **1.000** | 0.353 | 0.640 | 0.296 |
| Inductive Miner (0.2) | 0.996 | 0.540 | 0.644 | 0.290 |
| Inductive Miner (0.5) | 0.984 | 0.712 | 0.657 | 0.266 |
| Inductive Miner (0.6) | 0.982 | 0.710 | 0.659 | 0.262 |
| Inductive Miner (0.8) | 0.978 | 0.705 | 0.660 | 0.255 |
| Heuristic Miner | 0.907 | **0.795** | 0.591 | 0.228 |
| Alpha Miner | 0.550 | 0.040 | **1.000** | **0.302** |

Table 4.13: Performance metrics comparison **on the augmented dataset**. Inductive Miner shown with different noise thresholds, compared to baseline Heuristic and Alpha miners.

### 4.4.4 Final considerations

As we can see, the performances are not as good as in the first experiment, given the noiseness of the dataset, and also the limited part of the dataset provided to the LLMs. With respect to the augmented dataset benchmark, we see that the algorithms have better performances, however they are not reflected on the real scenario. The overall pattern is clear: the Heuristic Miner performs better than the Inductive Miner if we consider the precision, however from the point of view of the generalization, Alpha Miner and sometimes Inductive Miner are better.

# 5 | LLM Analysis

In this chapter we will see how LLMs are capable of analyzing the dataset and provide information about it. We will try to leverage the "reasoning" capabilities of these tools using a less structured prompt compared to the one used in the previous chapter. We will see how the responses can be insightful and provide useful information about the dataset, or to have a brief overview of the data.

**Prompt**

Here is an activity recognition dataset. Do a full analysis. Find any anomalies, bottlenecks, and special things.

## GPT-4o

**Response**

The dataset contains 2633 rows and 3 columns:

- Timestamp: date and time of the event (string format)
- CaseID: an identifier of the activity or device involved.
- Activity: description of the activity, which includes both device ID and activity type.

The data have no null values, which is good.

**Temporal Analysis and Activity Distribution.**

- Time range: the dataset covers the period from March 27, 2003 to April 11, 2003.
- Number of unique activities: there are 510 distinct activities.
- Most frequent activities:
    - Hot water washing (Toileting): 70 times
    - Showering (Bathing): 65 times
    - Hot water sink (Grooming): 63 times
    - Cold water sink (Grooming): 56 times
    - Toilet flush (Toileting): 50 times

**Anomalies Identified**

- Abnormal Transition Times.
    - 367 events have out-of-normal time intervals between activities.
    - Some examples:
        - Cabinet_80 ON Preparing_breakfast: interval of 331 seconds (5.5 minutes).
        - Medicine_cabinet_57 ON None: interval of 1362 seconds (22.7 minutes),

which is odd for an activity related to a medicine cabinet.
    - Door_140 ON Going_out_to_work: interval of 279 seconds (4.6 minutes), which could indicate a delay before leaving home.
- Activities with few occurrences (suspicious or rare).
  - 374 activities occur less than 5 times in the dataset.
  - Some examples:
    - Microwave_143 ON None (4 times): activities without a clear category.
    - Cabinet_53 ON Preparing_breakfast (4 times): could be an incomplete record.
    - Laundry_Dryer_90 ON Preparing_a_beverage (1 time): seems out of context, which could indicate a recording error.

**Identified Bottle Necks.**
- Critical threshold: more than 5 events per minute is considered an overload.
- Peak moments:
  - March 28 20:51 → 12 events in one minute.
  - March 29 15:19 → 12 events in one minute.
  - April 3 09:13 → 12 events in one minute.
  - April 10 19:16 → 12 events in one minute.

These times could indicate periods of intense activity or sensor malfunctions.

## Deepseek-R1

Considerations

Given the context window limitation, Deepseek have seen only a part of the CSV, and not the entire CSV and XES as GPT-4o.
Here it is interesting to also see the reasoning phase done by the model, which is not present in the GPT-4o response

Thinking

Okay, let's dive into this dataset. First, I need to understand the structure. There are three columns: Timestamp, CaseID, and Activity. The Timestamp is the date and time of the event, CaseID seems to represent specific objects or devices (like Cabinet_67, Toilet_Flush_100, etc.), and Activity describes the action being performed (e.g., Toileting, Preparing_breakfast).
Starting with basic exploration, I'll check the range of dates. The earliest entry is 2003-03-27 and the latest is 2003-04-06. So, the data spans about 11 days. Next, looking at the unique activities: Toileting, Preparing_breakfast, Bathing, Dressing, Preparing_lunch, Washing_dishes, Grooming, Going_out_to_work, Preparing_a_beverage, Cleaning, Lawnwork, Watching_TV, Going_out_for_shopping, Putting_away_groceries, Preparing_dinner, Preparing_a_snack, Doing_laundry, Putting_away_laundry, Going_out_for_entertainment, Washing_hands, Putting_away_dishes, and more. That's a variety of daily activities.

Now, checking for anomalies. One thing that stands out is some activities labeled as "None". For example, "Drawer_71 ON None" on 2003-03-27 06:46:12. That might indicate missing or unclassified activities. Another thing is multiple entries with the same timestamp and CaseID but repeated activities, like "Toilet_Flush_100 ON Toileting" twice at 2003-03-28 12:31:01. Could be duplicates or sensor errors.

Looking at bottlenecks. Preparing_lunch appears frequently and sometimes has many consecutive entries, like on 2003-03-29 from 14:32 to 15:14, there's a long sequence of Microwave_143 being turned on repeatedly. That might indicate inefficiency or prolonged activity time. Similarly, Washing_dishes on 2003-04-02 from 11:38 to 11:40 has numerous Dishwasher_70 activations, suggesting possible issues with the dishwasher or process delays.

Special things: There's an entry with "Oven_129 ON Dressing" on 2003-03-28 19:09:26. Using an oven for dressing seems odd. Maybe a mislabeling or sensor mix-up. Also, "Burner_94 ON Lawnwork" on 2003-03-29 13:24:32. Burners are typically for cooking, not lawnwork. Could be a misclassification or unusual activity.

Temporal patterns: Most activities like Preparing_breakfast, lunch, and dinner align with typical meal times. Toileting and Grooming are spread throughout the day, which is normal. However, some activities like Doing_laundry at odd hours (e.g., 2003-04-04 06:10:14) might indicate non-standard routines.

Device usage: Cabinet_67 and Toilet_Flush_100 are frequently used. Medicine_cabinet_57 and 58 are often involved in Toileting and Grooming. Microwave_143 and Refrigerator_91 are common in meal preparations. Door_140 is frequently associated with Going_out_to_work or other outings.

Potential issues: Repetitive activations of the same device in a short span (e.g., Microwave_143 multiple times in minutes) could indicate inefficiency or malfunction. Also, activities with "None" need clarification as they might represent unrecorded or erroneous data.

Seasonality: The data is over 11 days, so weekly patterns might emerge. For example, increased activity on weekends versus weekdays. However, without more days, it's hard to confirm.

Outliers: Entries like "DVD_56 ON Watching_TV" on 2003-03-28 19:55:24 make sense, but "Oven_129 ON Dressing" is puzzling. Need to check if these are genuine or errors.

In summary, anomalies include activities labeled "None", repetitive device usage, and mismatched device-activity pairs. Bottlenecks are seen in prolonged meal preps and dishwashing. Special cases involve unusual device-activity associations. Further steps would involve statistical analysis, visualizing activity durations, and checking for sensor reliability.

Response

**Analysis of MIT Activity Recognition Dataset**

**Dataset Overview**

- **Timeframe**: March 27, 2003, to April 6, 2003 (11 days).
- **Key Columns**:
  - **Timestamp**: Records activity start times.

  - **CaseID**: Devices/objects (e.g., `Cabinet_67`, `Microwave_143`).
  - **Activity**: Associated tasks (e.g., `Toileting`, `Preparing_lunch`).

**Key Observations**

**Anomalies**

1. **Unclassified Activities** (`Activity = "None"`):
   - 58 entries (e.g., `Drawer_71 ON None`, `Freezer_137 ON None`). Likely sensor errors or missing labels.
2. **Repetitive Entries**:
   - Duplicate timestamps for the same device (e.g., `Toilet_Flush_100 ON Toileting` twice at `2003-03-28 12:31:01`).
3. **Mismatched Device-Activity Pairs**:
   - `Oven_129 ON Dressing` (2003-03-28 19:09:26).
   - `Burner_94 ON Lawnwork` (2003-03-29 13:24:32). Likely mislabeling or sensor cross-talk.

**Bottlenecks**

1. **Prolonged Meal Preparation**:
   - `Preparing_lunch` on March 29 involved 50+ entries over 2 hours, with repeated `Microwave_143` activations (14:32-15:14), suggesting inefficiency.
   - `Preparing_dinner` on March 28 had 15+ `Refrigerator_91` activations, indicating frequent ingredient retrieval.
2. **Washing Dishes**:
   - On April 2, `Dishwasher_70` was activated 20+ times in 2 minutes (11:39-11:40), implying potential workflow issues.

**Special Patterns**

1. **High-Frequency Devices**:
   - `Cabinet_67` (toileting/grooming), `Toilet_Flush_100` (toileting), and `Refrigerator_91` (meal prep) are most active.
   - `Door_140` is strongly linked to `Going_out_to_work` and `Going_out_for_shopping`.
2. **Unusual Activity Times**:
   - `Doing_laundry` at odd hours (e.g., April 4 at 06:10:14).
3. **Repeated Sensor Triggers**:
   - `Shower_faucet_93` activated 10+ times in 5 minutes during `Bathing` (March 28 18:39-18:45).

**Temporal Insights**

- **Meal Preparation Peaks**: Align with typical times (e.g., breakfast ∼6:00-9:00, lunch ∼11:00-13:00).
- **Toileting/Grooming**: Distributed throughout the day, with clusters in mornings and evenings.
- **Laundry and Cleaning**: Often occurs midday or late at night (e.g., March 30 at 07:36:58).

**Recommendations**

1. **Sensor Validation**:

- Investigate devices with `Activity = "None"` and mismatched labels (e.g., `Oven_129 ON Dressing`).
- Check for sensor malfunctions causing duplicates (e.g., repeated `Microwave_143` triggers).

2. **Process Optimization**:
   - Streamline meal prep workflows to reduce repeated device interactions.
   - Analyze dishwashing bottlenecks (e.g., excessive `Dishwasher_70` usage).
3. **Activity Labeling**:
   - Clarify ambiguous activities (e.g., `Lawnwork` involving `Burner_94`).

**Conclusion** The dataset reveals patterns in daily routines but highlights inefficiencies in meal prep and cleaning. Anomalies suggest sensor mislabeling or errors. Addressing these issues could improve activity recognition accuracy and user experience.

## Qwen2.5-Max

Response

**Overview of Activities**

The dataset captures various activities performed in a smart home environment. Key activities include:

- **Toileting**: Involves sensors like `Toilet_Flush_100`, `Sink_faucet_hot_68`, and `Medicine_cabinet_57`.
- **Preparing Breakfast/Lunch**: Associated with kitchen appliances like `Microwave_143`, `Refrigerator_91`, and `Cabinet_55`.
- **Bathing/Dressing**: Triggered by sensors such as `Shower_faucet_93` and `Drawer_82`.
- **Other Activities**: Watching TV, doing laundry, grooming, etc.

**Temporal Patterns**

- Daily routines are evident, with morning activities (**Toileting**, **Preparing Breakfast**) followed by midday tasks (**Preparing Lunch**, **Washing Dishes**).
- Some activities (e.g., **Doing Laundry**) span extended periods, indicating multi-step processes.

**Sensor Usage Patterns**

- Frequently used sensors: `Cabinets`, `Drawers`, and kitchen appliances.
- Rarely used sensors: `DVD_56`, `Jewelry_box_139`, and `Laundry_Dryer_90`.

**Anomalies**

- Repeated activations of sensors like `Toaster_131` may indicate malfunctions or habitual behavior.
- Inconsistent activity labels (e.g., `None` during active events) suggest labeling issues.
- Unusual timing for certain activities (e.g., late-night bathing) may warrant further investigation.

**Bottlenecks**

- Overlapping activities (e.g., **Doing Laundry** and **Toileting**) complicate activity recognition.
- High dependency on specific sensors for critical activities increases vulnerability to sensor failures.

**Special Observations**

- Sequential patterns are observed, such as **Toileting** preceding **Bathing**.
- Nighttime activities (e.g., **Toileting**, **Grooming**) may indicate sleep disturbances.

**Recommendations**

- Investigate repeated activations and standardize activity labels.
- Optimize sensor placement and streamline overlapping activities.
- Use machine learning models to predict activity sequences and identify deviations.

## Llama3.2:3B

Response

Activity Recognition Dataset Analysis

The given dataset contains activity recognition data from a home environment, capturing various activities such as toileting, grooming, and others. Here's an in-depth analysis of the dataset:

Data Distribution

The dataset has a total of 2633 rows and 3 columns: Timestamp, CaseID, and Activity.

- Timestamp: The timestamps are in the format 'YYYY-MM-DD HH:MM:SS', indicating that the data was collected over a period of several days.

- CaseID: Each row corresponds to a unique case identifier, which can be used for individual or household-level analysis.

- Activity: This column contains various activity types, including toileting, grooming, and others.

Anomalies

- Duplicate Records: There are some duplicate records in the dataset, such as Medicine_cabinet_57 having multiple identical entries with different timestamps (2631 and 2632).

- Missing Timestamps: Some rows have missing or incomplete timestamps (e.g., Drawer_71). These might be due to errors in data collection or entry.

- Inconsistent Activity Mapping: In some cases, the activity is not clearly mapped to a specific action (e.g., "ON Toileting" for multiple activities like toilet flushing and light switch operation).

Bottlenecks

- Data Cleaning: The dataset requires thorough cleaning to handle duplicate records, missing timestamps, and inconsistent activity mapping.

- Feature Engineering: Additional features might be necessary to enhance the analysis or modeling capabilities of the data.

Special Things

Activity Types: The dataset includes various activities such as toileting, grooming, and others, which can be useful for understanding human behavior in different contexts.
Timestamp Clustering: By grouping timestamps by CaseID and Activity, it is possible to identify patterns or clusters in activity sequences (e.g., consecutive activities like "ON Toileting").
Case-Level Analysis: The dataset allows for analysis of individual cases (CaseID) separately from household-level analysis.
Data Cleaning
To address the issues mentioned above, the following steps can be taken:
- Remove duplicate records: Use the drop_duplicates method to remove duplicates.
- Fill missing timestamps: Use imputation techniques or interpolation methods to fill in missing timestamps.
- Standardize activity mapping: Map activities consistently and create new variables for each specific action.
Feature Engineering
To further enhance the analysis, additional features can be engineered:
- Activity sequences: Create a new feature by grouping consecutive activities together (e.g., "ON Toileting" followed by another toileting activity).
- Time intervals: Calculate time intervals between different activities to understand the duration and frequency of activities.
Modeling
The dataset can be used for various machine learning modeling tasks, such as:
- Classification models: Train models to predict activity types based on features extracted from the data.
- Clustering algorithms: Apply clustering techniques to group similar activities together.
- Time-series analysis: Analyze time-series data to identify patterns and trends in activity sequences.