Process Mining and Management Project

Davide Donà - 264467 Andrea Blushi - 264468

February 2026

Contents

1 Introduction

In modern business processes, the ability to predict outcomes and provide recommendations based on historical data is crucial for improving efficiency and supporting informed decision-making. This project investigates the use of process monitoring and machine learning techniques to analyze a real-life production log and to predict whether a process instance will lead to a positive outcome (fast trace: cycle-time < avg-cycle-time) or a negative outcome (slow trace). When a negative outcome is predicted, the approach generates recommendations that suggest modifications in the sequence of activities in order to steer the process towards a positive result. The effectiveness of the recommendation system is then evaluated by comparing the predicted outcomes with the actual outcomes in a held-out test dataset.

2 Implementation

A recommendation system based on a decision tree classifier was implemented to analyze production event logs. The implementation was developed in Python, leveraging the *PM4Py* library for process mining tasks and the *scikit-learn* for machine learning functionality. The overall execution workflow is encapsulated in the *notebook.ipynb* file, which orchestrates the execution of all supporting utility functions.

2.1 Encoding the Event Log

The event log, provided in *XES* format, was imported using the *PM4Py* library and converted into a *PM4Py* event log object suitable for subsequent processing. A pruned version of the event log was then constructed by considering only trace prefixes. Specifically, each trace was truncated to a fixed number of events defined by the parameter *prefix_length*. Since ongoing traces in real-world scenarios are inherently incomplete, the classifier was trained exclusively on such prefixes.

Subsequently, the set of unique activity names was extracted from the complete event log and used to define the feature space. In addition to the activity-based features, each encoded trace included a trace identifier, the prefix length, and a ground truth label indicating whether the trace outcome was positive or negative.

To enable training with a decision tree classifier, the pruned traces were encoded using a *Boolean encoding* representation. For each unique activity, a binary feature was created indicating whether the activity occurred at least once within the trace prefix. This encoding scheme results in a fixed-dimensional feature space and avoids the need to pad traces to a uniform length. The final outcome of this step is a tabular *DataFrame* in which each row corresponds to a trace

prefix and each feature column represents the presence or absence of a specific activity.

2.2 Training the Decision Tree Classifier

Following the encoding of the pruned event log, a decision tree classifier was trained using the *scikit-learn* library. The encoded dataset was partitioned into a feature matrix $\boldsymbol{X}$, obtained by excluding the trace identifier and label columns, and a label vector $\boldsymbol{y}$ containing the ground truth outcomes. The original *XES* file provided a predefined split between training and test sets, which was preserved during this process.

After training the classifier, the learned decision rules were examined by visualizing the resulting decision tree using the visualization utilities provided by *scikit-learn*.

2.3 Optimizing the Classifier Hyperparameters

To improve classification performance, hyperparameter optimization was conducted using the *Hyperopt* library. This tool systematically explores a predefined search space, iteratively testing combinations of critical hyperparameters—such as *max_depth*, *max_features*, and the split criterion (e.g., *Gini* or *entropy*)—to identify the optimal configuration. For each configuration, the model was trained and evaluated using the F_1-score as the primary performance metric. The hyperparameter configuration yielding the highest F_1-score was selected as the final model. In addition, the search space for *max_depth* was restricted to comparatively small values in order to limit model complexity and reduce the risk of overfitting on the training set.
To ensure reproducibility, a fixed random seed was set for all stochastic processes involved in the optimization and training phases.

2.4 Generating Predictions

The optimized classifier was subsequently applied on the test set to generate predictions. The predicted labels were obtained by applying the model to the feature set derived from the encoded test log. Similar to the training phase, the features ($\boldsymbol{X}_{test}$) were constructed by removing the trace identifier and ground truth label columns. The classifier's performance was subsequently evaluated using standard scikit-learn metrics, including *Accuracy*, *Precision*, *Recall*, and the *F1-score*.

2.5 Extracting Recommendations

Due to the inherent interpretability of decision tree models, the trained classifier can be used to derive proactive recommendations for traces predicted to result

in a negative outcome. The recommendation generation procedure consists of three main steps:

1. **Extracting Positive Paths**: identifying all distinct decision paths leading to leaf nodes associated with a positive outcome.
2. **Filtering Compliant Paths**: selecting, for a given negatively predicted trace prefix, those positive paths whose conditions are not violated by the activities already observed.
3. **Generating Recommendations**: choosing the compliant positive path with the highest confidence score and determining the activities required to complete that path.

It should be noted that the implemented function signature differs from the originally proposed one. In particular, the *class_values* parameter was removed to simplify the implementation, as the experimental setting is restricted to binary classification.

Extracting Positive Paths from the Decision Tree: To extract all positive paths from the decision tree, the decision tree is traversed from the root node using a *depth-first search* (DFS) strategy. At each internal node, the corresponding feature, comparison operator (e.g., $\leq$, $>$), and threshold value were recorded. Upon reaching a leaf node labeled with a positive outcome, the complete sequence of conditions along the path, together with its associated confidence score, was stored.

To prevent the selection of statistically unrepresentative paths, a weighted confidence score was employed. Let n_{leaf} denote the number of training samples reaching a given positive leaf, n_{root} the number of samples at the root node, and p_{pos} the proportion of positive samples at that leaf. The confidence score s associated with a positive path is defined as:

$$s \;=\; p_{\text{pos}} \cdot \frac{n_{\text{leaf}}}{n_{\text{root}}}.$$

This definition combines the local confidence of the leaf (via p_{pos}) with its global context (via $n_{\text{leaf}}/n_{\text{root}}$). This penalizes paths that achieve high purity based on a negligible number of observations, thereby prioritizing paths that are both accurate and statistically significant.

The Algorithm 2.5 formally defines the procedure for extracting positive paths from the decision tree.

Algorithm 1 Get Positive Paths

```
1: function GetPositivePaths(node, current_path, positive_paths)
2:     if node is leaf then
3:         if node.label == positive then
4:             Add (current_path, node.confidence) to positive_paths
5:         end if
6:     else
7:         Append (node.feature, ≤, node.value) to current_path
8:         Call GetPositivePaths(node.left, current_path, positive_paths)
9:         Remove last element from current_path
10:
11:        Append (node.feature, >, node.value) to current_path
12:        Call GetPositivePaths(node.right, current_path, positive_paths)
13:        Remove last element from current_path
14:    end if
15: end function
```

Filtering Compliant Paths: For each trace prefix predicted as negative, the set of extracted positive paths was filtered to retain only those that are compliant with the observed activities. A path is defined as compliant if none of its feature conditions are contradicted by the trace prefix, that is, if every condition along the path is either satisfied or refers to a feature that does not yet appear in the prefix. Among all compliant paths, the one with the highest confidence score was selected. In case of a tie in confidence, the shortest path was chosen as a secondary selection criterion.

Generating Recommendations: Once the most suitable positive compliant path was identified, recommendations were generated by determining which activities needed to be added to the current trace in order to satisfy the conditions of that path. For each test-set prefix, the recommendation set is defined as follows:

- *None*, if the trace was already predicted to result in a positive outcome;
- the empty set, if no compliant positive path could be identified;
- otherwise, the set of activities corresponding to the unsatisfied conditions along the selected positive path.

2.6 Evaluation of Recommendations

The quality of the generated recommendations was evaluated by comparing them against the complete traces in the test set, which serve as ground truth. Each trace in the test set was matched with its corresponding prefix and the recommendation generated for that prefix.

Recommendations for which no action was required were excluded from the evaluation. Recommendations consisting of an empty set were considered *not followed*, reflecting the absence of a feasible corrective action. A recommendation was considered *followed* if and only if all Boolean conditions associated with the selected path were satisfied in the full trace. If any condition was violated, the recommendation was classified as *not followed*.

The recommendation outcome (followed or not followed) was then compared with the ground truth label of the trace to derive an approximate confusion matrix defined as follows:

- **True Positives**: the recommendation was followed in the actual trace and the ground truth outcome is positive.
- **False Positives**: the recommendation was followed in the actual trace but the ground truth outcome is negative.
- **True Negatives**: the recommendation was not followed in the actual trace and the ground truth outcome is negative.
- **False Negatives**: the recommendation was not followed in the actual trace but the ground truth outcome is positive.

Based on this confusion matrix, standard evaluation metrics such as accuracy, precision, recall, and F1-score were computed to assess the effectiveness of the generated recommendations.

3 Evaluation

This section presents the evaluation results for both *prefix_length* = 5 and *prefix_length* = 11 of the decision tree classifier and the recommendation system built on top of it.

3.1 Hyperparameters

Tables 1 and 2 report the optimized hyperparameters obtained through *Hyperopt* for both prefix lengths.

For *prefix_length* = 5, the optimizer selects a shallow tree with a maximum depth of 3 and a small number of features considered at each split. This choice limits overfitting in a setting where only partial information about the trace is available, while still allowing the model to exploit the most informative activities. The use of the entropy criterion leads to more balanced and interpretable decision paths.

For *prefix_length* = 11, the optimal configuration still constrains the maximum depth to 3, confirming that deeper trees do not provide additional benefits even

when more information is available. However, the maximum number of features increases, indicating that longer prefixes allow the model to effectively leverage a richer set of activities. In this case, the Gini criterion is selected, suggesting that it better captures class separation when more contextual information is present.

Hyperparameter	Value
Criterion	Entropy
Max Depth	3
Max Features	2

Table 1: Optimized hyperparameters for the decision tree classifier for *prefix_length* = 5.

Hyperparameter	Value
Criterion	Gini
Max Depth	3
Max Features	3

Table 2: Optimized hyperparameters for the decision tree classifier for *prefix_length* = 11.

3.2 Decision Tree Structure

Figures 1 and 2 illustrate the structure of the trained decision trees for the two prefix lengths. Leaf nodes are color-coded (blue for positive outcomes, orange for negative outcomes) to indicate prediction distributions. The controlled maximum depth and limited number of features ensure interpretability, preventing excessive branching. The tree structures provide valuable insights into which activities are most discriminative for predicting case outcomes, making the model transparent.

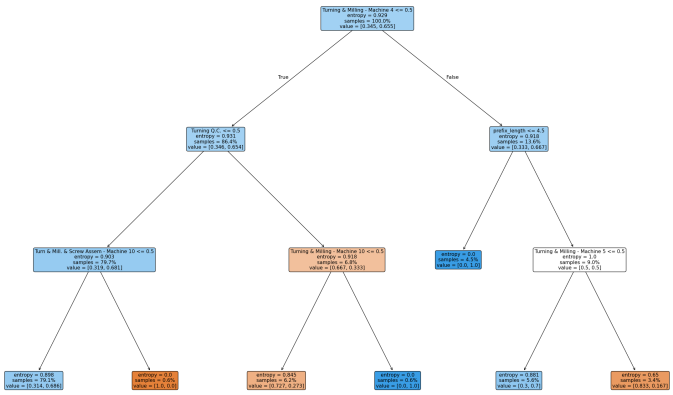

Figure 1: Visualization of the trained decision tree classifier for *prefix_length* = 5. Each node represents a decision based on the presence or absence of specific activities, leading to recommendations at the leaf nodes.

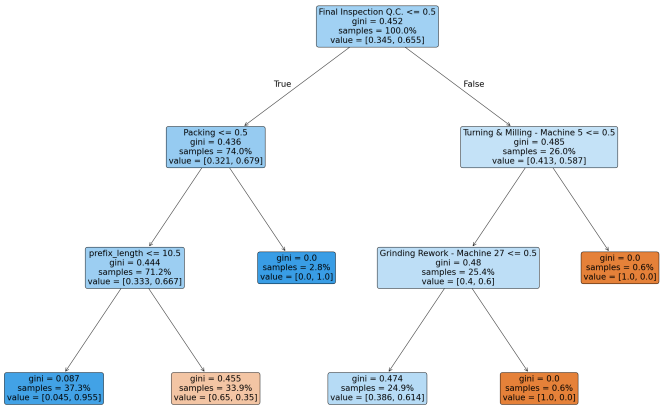

Figure 2: Visualization of the trained decision tree classifier for *prefix_length* = 11.

3.2.1 Evaluation Metrics

The predictive performance of the decision tree classifiers is summarized in Tables 3 and 4.

For *prefix_length* = 5, the model exhibits moderately balanced performance, capturing meaningful patterns despite limited information in the trace prefixes. Increasing the prefix length to *prefix_length* = 11 provides additional contextual information, enabling the model to make more confident positive predictions. These results confirm that increasing the prefix length generally improves outcome prediction, as more activities contribute to defining the process trajectory. Confusion matrices in Figures 3 and 4 provide further insight into the distribution of predictions and misclassifications.

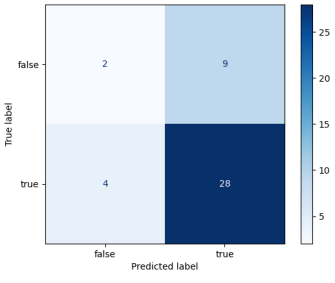

Figure 3: Confusion matrix of predictions in the test set (*prefix_length* = 5).

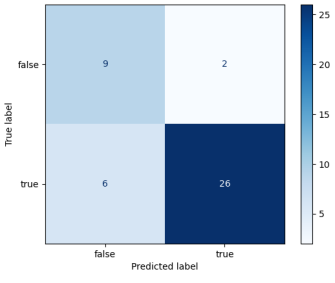

Figure 4: Confusion matrix of predictions in the test set (*prefix_length* = 11).

Metric	**Value**
Accuracy	69.77%
Precision	64.84%
Recall	69.77%
F1-Score	66.42%

Table 3: Evaluation metrics for the decision tree classifier on the test set for *prefix_length* = 5.

Metric	**Value**
Accuracy	81.40%
Precision	84.45%
Recall	81.40%
F1-Score	82.21%

Table 4: Evaluation metrics for the decision tree classifier on the test set for *prefix_length* = 11.

3.3 Recommendation Analysis

Recommendation Generation To illustrate recommendation extraction, a simplified decision tree is considered (Figure 5).

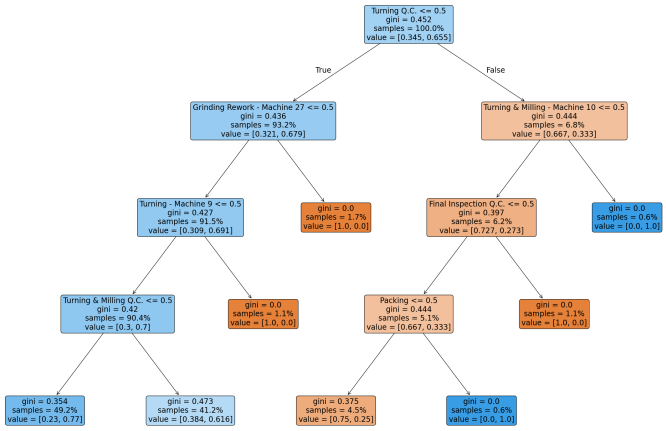

Figure 5: Simplified decision tree used to illustrate the recommendation extraction process. Given a node, the left child branch is taken if the condition is true (activity is absent in the trace), while the right child branch is taken if the condition is false (activity is present in the trace).

Applying the extraction of positive paths (Algorithm 2.5) to this tree, we obtain the following positive paths with their confidence scores:

- Path 1: *Turning Q.C.* = F $\wedge$ *Grinding Rework - Machine 27* = F $\wedge$ *Turning - Machine 9* = F $\wedge$ *Turning & Milling Q.C.* = F with confidence 0.77;
- Path 2: *Turning Q.C.* = F $\wedge$ *Grinding Rework - Machine 27* = F $\wedge$ *Turning - Machine 9* = T with confidence 0.80;
- Path 3: *Turning Q.C.* = T $\wedge$ *Turning & Milling - Machine 10* = F $\wedge$ *Final Inspection Q.C.* = F $\wedge$ *Packing* = T with confidence 1.0;
- Path 4: *Turning Q.C.* = T $\wedge$ *Turning & Milling - Machine 10* = T with confidence 1.0;

Given a trace prefix $\sigma = \{$*Final Inspection Q.C.* $= T$, *Packing* $= T$, *Grinding Rework - Machine 27* $= T\}$ predicted as negative, compliant paths are identified as follows: Given a trace prefix $\sigma = \{$*Final Inspection Q.C.* $= T$, *Packing* $= T$, *Grinding Rework - Machine 27* $= T\}$ predicted as negative, compliant paths are identified as follows:

- Paths 1 and 2 are non-compliant, violating *Grinding Rework - Machine 27*=F.
- Path 3 is compliant.
- Path 4 is non-compliant, violating *Final Inspection Q.C.*=F.

Given this, the recommendations for the given prefix trace are the activities that need to be added to follow Path 3. Since both *Turning Q.C.* and *Turning & Milling - Machine 10* need to be true to follow Path 3, the recommended activities will be those. This is illustrated in Figure 6, with the green dotted nodes.

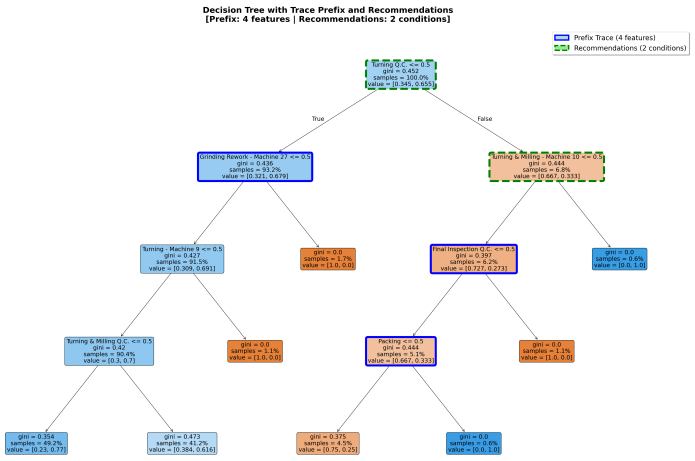

Figure 6: The blue squared nodes represent the activities present in the given prefix trace. The green highlighted nodes represent the recommended activities.

Recommendations Quality Tables 5 and 6 present the evaluation of our recommendation system. Metrics are approximations, as the ground truth is based on actual traces rather than hypothetical scenarios in which recommendations were followed.

For *prefix_length* = 5, all metrics are 0%, indicating that the few generated recommendations were not followed. This is likely due to insufficient contextual information in very short prefixes, making it difficult for the system to provide effective suggestions.

For *prefix_length* = 11, the recommendation system demonstrates substantial improvement across all metrics compared to shorter prefix lengths. This demonstrates that a sufficient prefix length is required for the recommendation system to reliably identify actionable patterns and align them with successful outcomes.

Metric	Value
Accuracy	0.00%
Precision	0.00%
Recall	0.00%
F1-Score	0.00%

Table 5: Evaluation metrics for the recommendation system on the test set for *prefix_length* = 5.

Metric	Value
Accuracy	46.67%
Precision	60.00%
Recall	33.33%
F1-Score	42.86%

Table 6: Evaluation metrics for the recommendation system on the test set for *prefix_length* = 11.

Figures 7 and 8 present the corresponding confusion matrices, providing further insight into recommendation outcomes.

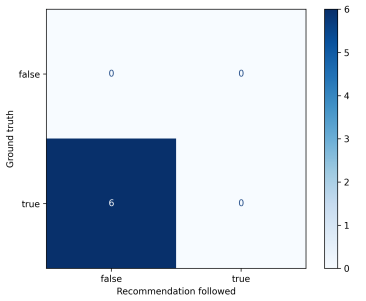

Figure 7: Confusion matrix of recommendations (*prefix_length* = 5).

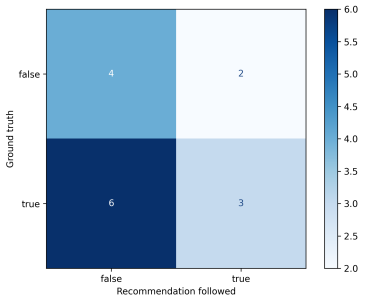

Figure 8: Confusion matrix of recommendations (*prefix_length* = 11).

3.4 Performance by Prefix Length

Figure 9 compares the F1-score for the decision tree classifier and the recommendation system across different prefix lengths. The decision tree classifier maintains relatively stable performance, whereas the recommendation system exhibits marked improvement as prefix length increases. This trend highlights that, while outcome prediction can remain effective with short prefixes, meaningful recommendation generation requires a minimum threshold of contextual information.

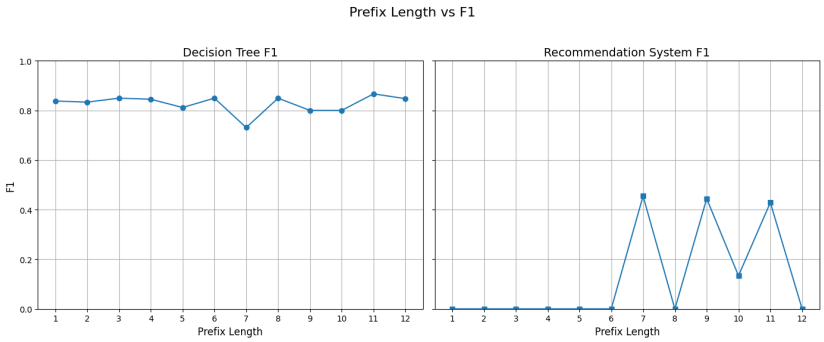

Figure 9: The box plots illustrate the distribution of F1-scores across different prefix lengths for both the decision tree classifier and the recommendation system.

4 Conclusion

Given the limited number of samples in our dataset, the results should be considered preliminary. Nevertheless, they are promising and suggest that, with the availability of more data, the proposed recommendation system could become a valuable tool for process improvement. Our evaluation indicates that the effectiveness of recommendations is highly dependent on the amount of process context available. Longer prefixes capture more detailed information about the process execution, enabling the recommendation system to generate more accurate and relevant suggestions.

Overall, this project demonstrates the feasibility of combining process mining and machine learning techniques to predict process outcomes and generate actionable recommendations. With further refinement and larger datasets, such systems have the potential to significantly enhance business process management and optimization.