university of groningen

faculty of science and engineering

Adaptive Business Process Analysis Using Machine Learning Algorithms

Radu Andrei Sterie

university of groningen

faculty of science and engineering

University of Groningen

Bachelor's Thesis

To fulfill the requirements for the degree of
Bachelor of Science in Computer Science
at University of Groningen under the supervision of
Prof. dr. Dimka Karastoyanova (Computer Science, University of Groningen)
and
Arash Yadegari (Computer Science PhD student, University of Groningen)

Radu Sterie (s4151615)

August 10, 2023

Contents

Acknowledgments

I would want to convey my appreciation to everyone who helped me finish my academic education and thesis. Their inspiration and assistance have had a significant impact on both this project and my entire academic career.

Arash, my supervisor, deserves my sincere gratitude, so I will start there. Your expertise and dedication have been crucial to the success of this research effort. Your suggestions and constructive criticism enabled me to raise my standards. I appreciate the opportunity to work with you.

I also want to thank my dear friend Dyllan deeply for everything. Your ongoing motivation, assistance, and intelligent discussions have been an ongoing resource of motivation for me. Your willingness to listen, offer informed debates, and give constructive criticism has been essential in defining the direction of this thesis. I am very appreciative that you stood by my side throughout this process.

I also want to express my gratitude to everyone who offered any kind of assistance to me while I was conducting my research. I'm grateful to my family for their encouragement, unconditional love, and confidence in my skills. My academic attempts rely on the support that you provided. I also want to thank my friends for their intelligent criticism, stimulating discussions, and moral support along this journey. Your presence has enhanced the experience and made it even more enjoyable.

Furthermore, I want to express my gratitude to the University of Groningen's faculty and staff for creating a research-friendly atmosphere. Your dedication to promoting knowledge and providing helpful resources has been essential.

Last but not least, I would want to express my gratitude to everyone who has contributed to this area of study, whether directly or indirectly. Academics, practitioners, and researchers worked together to build the framework for this thesis.

ML.LOG

Abstract

The purpose of this undergraduate research project is to examine how machine learning techniques may be used to find correlations in adaptive business process logs. The project employs an innovative methodology to analyze business process data using the Generalized Learning Vector Quantization (GLVQ), K-Nearest Neighbors (KNN), Decision Trees, and Random Forest algorithms. In order to acquire practical insights into the innate patterns inside the business processes, it is intended to extract relevant adaptations (or rules) from these logs.

In this study, two specifically created programs were used: >_NEXT(LOG), which creates logs in accordance with predetermined rules, and ML.LOG, a machine learning module that extracts these rules from the generated logs. To achieve the optimum fit for each algorithm, the machine learning models were trained using a number of different parameters. Metrics including the F1 score, recall, and accuracy were used to rate the models. In addition to being saved for potential use in the future, the data were displayed to aid with comprehension.

The research discovered that different models had varying degrees of effectiveness in extracting rules from the adaptive business process logs, with Decision Tree and Random Forests performing the best. By offering a methodology for using machine learning to extract useful insights from adaptive business process logs, this study makes a contribution to the expanding field of machine learning applications in business process analysis. This method has limitations, such as a limited number of available machine learning models, but it has promise for upcoming study and applications.

This thesis also covers the methodology's specifics, the models' performance outcomes, the rules that were retrieved, and the consequences of these findings. The findings are anticipated to spur additional research and advancements in the field of adaptive business process analysis.

1 Introduction

The analysis of business processes is a critical component in the optimization of operations within an organization. The advent of adaptive business processes, which are capable of changing in response to varying conditions, has introduced new complexities in their analysis. Traditional methods of business process analysis often struggle to effectively handle the dynamic nature of these adaptive processes, leading to a need for more advanced, automated techniques.

Machine learning, a subset of artificial intelligence, has emerged as a promising solution to this challenge. Its ability to learn patterns from large datasets and make intelligent, predictive decisions makes it particularly suited for analyzing adaptive business process logs. For instance, a study by Bozorgi et al. (2020) proposed an approach that combines process mining and causal machine learning to generate case-level recommendations from event logs, demonstrating the potential of machine learning in this context [1].

Furthermore, machine learning models have shown their adaptive learning capacities in various domains, such as telecommunication industries, where they have been used for extrapolative analysis of big data chunks (Isabona et al., 2022) [2]. These models can be adapted to analyze business process logs, providing valuable insights into the underlying processes.

However, the application of machine learning in business process analysis is not without its challenges. The black-box nature of some machine learning models, such as deep learning models, can limit their usefulness for what-if process analysis (Camargo et al., 2021) [3].

Given these challenges and opportunities, this thesis aims to explore the use of machine learning algorithms for the analysis of adaptive business process logs. The goal is to develop a system that can effectively identify correlations in these logs, thereby providing valuable insights into the underlying business processes.

This research project expands upon Arash Yadegari's work on business process model research. By leveraging Arash's framework as a starting point, this research extends his ideas to address emerging challenges, enhance model accuracy and efficiency, and emphasize practical applications and industry relevance. The findings presented here contribute to the ongoing development of business process modeling methodologies and provide valuable insights for researchers, practitioners, and organizations striving for operational excellence in today's dynamic business landscape. [4] [5]

1.1 Problem Statement

The primary problem this thesis aims to address is the identification of correlations in adaptive business process logs using machine learning algorithms. Despite the potential of machine learning in this context, its application in the analysis of adaptive business process logs is still a challenging task due to the dynamic nature of these processes and the complexity of the data they generate.

Specifically, the objectives of this thesis are:

1. To develop a system that can effectively analyze adaptive business process logs generated by the >_NEXT(LOG) program and identify patterns or rules within these logs.

2. To compare the performance of four different machine learning models (GLVQ, KNN, Decision Trees, and Random Forest) in identifying these patterns and determining the best model for this task.

3. To evaluate the performance of the selected model(s) using various metrics such as F1 score, recall, and accuracy, and to visualize the results for better understanding.
4. To enable the saving and loading of trained models for future use and comparison against new datasets.

The research questions that this thesis intends to answer are:

Q1. How effectively can machine learning algorithms identify correlations in adaptive business process logs?

Q2. Which machine learning model among GLVQ, KNN, Decision Trees, and Random Forest performs best in this task?

Q3. How can the performance of these models be accurately evaluated and compared?

Q4. How can the results be effectively visualized to provide meaningful insights into the underlying business processes?

By addressing these objectives and answering these research questions, this thesis aims to contribute to the field of adaptive business process analysis and machine learning, providing a tool that can help organizations better understand and optimize their business processes.

1.2 Scope and Limitations

The scope of this thesis is primarily focused on the application of machine learning algorithms for the analysis of adaptive business process logs. The research will involve the development and evaluation of a system that can effectively identify patterns or rules within these logs. The system will be tested using four different machine learning models: GLVQ, KNN, Decision Trees, and Random Forest. The performance of these models will be evaluated using various metrics and the results will be visualized for better understanding. However, there are several limitations to this study:

1. **Data Limitations**: The research is based on the analysis of adaptive business process logs generated by the >_NEXT(LOG) program. The findings may not be generalizable to other types of business process logs or to logs generated by different programs.
2. **Model Limitations**: While the study will compare the performance of four different machine learning models, there are many other models that could potentially be used for this task. The results of this study may not be applicable to these other models.
3. **Evaluation Limitations**: The evaluation of the models' performance is based on specific metrics such as F1 score, recall, and accuracy. These metrics may not capture all aspects of a model's performance, and other metrics could potentially yield different results.

Despite these limitations, this study aims to provide valuable insights into the application of machine learning algorithms for the analysis of adaptive business process logs, and to contribute to the ongoing development of tools and techniques for business process analysis.

2 Background Literature

The application of machine learning algorithms for the analysis of adaptive business process logs is a relatively new and rapidly evolving field of research. This literature review aims to provide an overview of the key studies and developments in this area, focusing on the use of machine learning techniques for business process analysis and the challenges and opportunities associated with analyzing adaptive business process logs.

The literature review is divided into two main sections. The first section, Business Process Analysis, reviews the existing literature on business process analysis, including traditional approaches and recent trends. It discusses the challenges and limitations of existing techniques in handling adaptive business process logs and highlights the need for more advanced, automated techniques.

The second section, Machine Learning in Business Process Analysis, surveys the literature on the application of machine learning algorithms in business process analysis. It identifies relevant studies that have used similar techniques or addressed related problems and discusses the strengths and weaknesses of the different machine learning algorithms in this context.

By reviewing the existing literature, this study aims to identify gaps in the current knowledge and provide a solid foundation for the subsequent analysis of adaptive business process logs using machine learning algorithms.

2.1 Business Process Analysis

Business process analysis is a critical component in the optimization of operations within an organization, particularly in the era of ”big data” where large amounts of data are collected and need to be analyzed in meaningful and scalable ways [6]. One of the key challenges in this field is the analysis of data that results from the execution of business processes, also known as event logs.

The field of process mining, a subset of business process analysis, is concerned with the automatic extraction of process models from event logs [7]. However, the representational bias, or the way processes are modeled, plays a crucial role in process mining. The Business Process Model and Notation (BPMN) standard is widely used in this context as it allows for the integration of control flow, subprocesses, data flows, and resources within one diagram, making it attractive for both process miners and business users [7].

2.2 Machine Learning in Business Process Analysis

Machine learning has emerged as a promising tool for business process analysis. One approach has been proposed to analyze an event log of a business process to generate case-level recommendations of treatments that maximize the probability of a given outcome [1]. This method employs action rule mining to pinpoint treatments that appear alongside the outcome in specific scenarios. Additionally, it utilizes causal machine learning to find groups of instances where a treatment significantly influences the outcome, even after accounting for other influencing factors. [1].

Another important aspect of applying machine learning to business process analysis is the issue of privacy. A particular study introduced a privacy-focused disclosure method. This method increases the number of cases in the log and introduces noise to the timestamps, ensuring a specific level of privacy protection. [8]. This approach allows for the analysis of event logs containing private information while complying with data protection regulations.

Another innovative approach in the realm of business process analysis is the utilization of autoencoders for detecting and analyzing anomalies in the execution of a business process. Autoencoders, a type of neural network, have been proposed as a method to detect anomalies without relying on prior knowledge about the process. This method is particularly advantageous as it can be trained on datasets that already contain anomalies. The primary advantage of this approach is its ability to discern which specific event within a sequence is anomalous, rather than flagging the entire sequence. Furthermore, it can pinpoint which attribute characteristic of an event is the root cause of the anomaly [9].

In a comparative evaluation against state-of-the-art anomaly detection methods, the autoencoder-based approach demonstrated superior performance. Specifically, in experiments, this approach achieved an F1 score of 0.87, outperforming the best unaltered state-of-the-art method, which reached an F1 score of 0.72 [9].

Thus, the application of autoencoders offers a novel and effective method for anomaly detection in business process data, providing detailed insights into the specific events and attributes causing the anomalies. This approach holds significant promise for enhancing business process analysis and ensuring optimal process execution [9].

Finally, the comparison of business process variants using event logs is a common use case in process mining. A new method has been introduced to identify notable differences between variants by examining complete process traces. This method employs a strategy to derive a mutual fingerprint, which is a directly-follows graph, from the event logs of both variants. This aids in comprehending the statistical variances in both the sequence of operations and efficiency aspects. [10].

3 Methods

The methodology employed in this study involves a combination of data generation, preprocessing, machine learning model selection, training, evaluation, and comparison. The overall process is designed to be iterative, allowing for the refinement of models based on their performance.

The first step in the methodology involves the generation of adaptive business process logs using the >_NEXT(LOG) program created by Dyllan Cartwright [11]. This program creates logs that follow certain rules, providing a dataset for the machine learning models to analyze.

Once the logs are generated, they are preprocessed to prepare them for analysis. This preprocessing step involves cleaning the data, handling missing values, and transforming the data into a format that can be used by the machine learning models.

The next step involves the selection of machine learning models for analysis. Four models are used in this study: Generalized Learning Vector Quantization (GLVQ), K-Nearest Neighbors (KNN), Decision Trees, and Random Forest. These models were chosen based on their suitability for the task and their ability to handle the type of data generated by the >_NEXT(LOG) program.

After the models are selected, they are trained using the preprocessed data. The training process involves tuning the parameters of each model to achieve the best performance. The models are then evaluated using various metrics, including F1 score, recall, and accuracy. These metrics provide a quantitative measure of the performance of each model, allowing for their comparison.

Finally, the models are compared against each other to determine which one performs best. This comparison involves analyzing the performance metrics of each model and considering the strengths and weaknesses of each approach.

This methodology provides a systematic approach to analyzing adaptive business process logs using machine learning. It allows for the exploration of different models and the refinement of these models based on their performance, ultimately leading to the identification of the most effective approach for this task.

3.1 Data Collection and Preprocessing

The data used in this study are generated using the >_NEXT(LOG) program, a custom-built software tool that creates adaptive business process logs. These logs are created based on specific rules, such as ”if event A's duration $>$ 100, skip event C”. This rule-based generation of logs provides a controlled environment for testing the machine learning models, as the underlying patterns in the data are known and can be used to validate the results.

The generated logs are saved in a CSV format, which is a widely used standard for storing tabular data and is compatible with many data analysis tools and libraries. Each row in the CSV file represents an instance of a business process, with columns representing different attributes of the process, such as the duration of events.

Once the data is collected, it undergoes a preprocessing step to prepare them for analysis. This step involves cleaning the data to remove any errors or inconsistencies that could affect the performance of the machine learning models.

In the context of preprocessing, it is necessary to eliminate events that occur after a certain rule condition to prevent the occurrence of one-to-one mappings. For instance, consider the removal of event ”C” based on the rule ”if the duration of event A exceeds 100, exclude event C.” The rationale behind this lies in the fact that when event A's duration surpasses 100, the duration of event C becomes 0, resulting in the instance label being assigned as 1 or "adapted". Consequently, the duration of event A becomes directly and uniquely mapped to the target variable, leading to a one-to-one relationship.

The preprocessing step also involves transforming the data into a format that can be used by the machine learning models. This may involve normalizing numerical variables to a standard scale, and reshaping the data into the input format required by the models.

This data collection and preprocessing methodology ensures that the data used in the study are reliable, clean, and in a suitable format for analysis. It provides a solid foundation for the subsequent steps of model selection, training, and evaluation.

3.2 Machine Learning Models

In this study, four machine learning models are used to analyze the adaptive business process logs: Generalized Learning Vector Quantization (GLVQ), K-Nearest Neighbors (KNN), Decision Trees, and Random Forest. These models were chosen based on their ability to handle the type of data generated by the >_NEXT(LOG) program and their suitability for the task of identifying patterns or rules within these logs.

GLVQ is a prototype-based supervised classification algorithm. It is particularly effective for classification tasks where the classes are not linearly separable. It works by defining prototypes for each class and then classifying new instances based on their similarity to these prototypes.

KNN is a simple yet powerful algorithm that classifies new instances based on their similarity to existing instances. It considers the 'K' closest instances to a new instance and assigns the most common class among these 'K' instances to the new instance.

Decision Trees is a type of model that makes decisions based on a series of questions about the features of the data. It is particularly useful for tasks where interpretability is important, as the decisions made by the model can be easily visualized and understood.

Random Forest is an ensemble learning method that combines multiple decision trees to make a final decision. It is known for its robustness and ability to handle large datasets with many features.

These models have been widely used in various fields and have shown their effectiveness in different tasks. For instance, Islam and Amin (2020) used Random Forest and Gradient Boosting Machine learning techniques for predicting products' backorder in the supply chain, achieving improved performance using a ranged approach when the dataset is highly biased with random error [12]. Similarly, Ahmed et al. (2022) surveyed machine learning techniques used for spam detection in email and IoT platforms, including Decision Trees and Random Forest [13]. Palša et al. (2022) used XGBoost and extremely randomized trees algorithms for malware detection, achieving 96.4% accuracy [14].

These studies demonstrate the versatility and effectiveness of the selected machine learning models, suggesting their potential suitability for the task of analyzing adaptive business process logs.

3.3 Training and Evaluation

The training of the selected machine learning models is carried out using the ML.LOG program. This program takes as input the preprocessed adaptive business process logs and the selected machine learning model, along with a set of parameters for the model. The parameters can be input manually, either as a list of parameters that are provided for the program to automatically search for the best parameters using Grid Search Cross Validation or simply as a single parameter. If values are not provided for the parameters, the default values will be used.

Grid Search Cross Validation is a parameter tuning technique that systematically works through multiple combinations of parameters, cross-validating as it goes to determine which combination gives the best performance [15]. The beauty of this technique is that it can test many combinations with a few lines of code.

After training the model, its effectiveness is gauged using various evaluation measures. These include the F1 score, recall, and accuracy. The F1 score evaluates the balance between a model's precision and recall, with 1 being the ideal score (indicating flawless precision and recall) and 0 being the least desirable. Recall, sometimes referred to as sensitivity, represents the percentage of relevant instances correctly identified by the model. Meanwhile, accuracy measures the proportion of predictions the model got right out of all its predictions.[16]

The experimental setup involves training and testing the models on different datasets. The datasets are split into a training set and a test set, with the training set used to train the model and the test set used to evaluate its performance. This setup allows for the assessment of how well the model generalizes to unseen data.

The software program also incorporates an "Auto-ML" functionality that facilitates the comparison and selection of different models. This feature involves an iterative process wherein each desired model is evaluated, and the best parameters are selected from the provided lists. Scores are generated for each model, and the one with the highest F1 score is identified as the optimal choice.

Furthermore, the program offers a valuable capability to choose between parsing a file generated by the >_NEXT(LOG) approach or a general file. In the case of a general file, the first $n-1$ columns of the CSV file represent the features, while the n^{th} column represents the target variable. This versatile feature empowers users to employ diverse datasets for a wide range of tasks, extending beyond the confines of Adaptive Business Process Analysis.

This rigorous training and evaluation process ensures that the selected machine learning models are well-suited for the task of identifying patterns in adaptive business process logs, and that their performance is accurately assessed and compared.

4 Implementation

4.1 Tools and Technologies

In the execution of this study, we utilized a range of tools and technologies to aid in the analysis and elucidation of decision-making processes. For the graphical representation of Random Forest and Decision Tree models, we employed Graphviz. This open-source tool for graph visualization was instrumental in visually depicting the structural data of these models, thereby simplifying the understanding of their decision-making mechanisms.

For the Generalized Learning Vector Quantization (GLVQ) and K-Nearest Neighbors (KNN) models, we used Matplotlib. This Python-based plotting library facilitated the production of static, animated, and interactive visualizations, which were crucial in deciphering the decision-making processes of these models.

Additionally, scikit-learn, a well-established machine learning library in Python, was used to implement and manipulate Decision Trees, KNN, and Random Forest models. This free software tool offers a suite of data mining and data analysis utilities, making it an ideal choice for our needs. It provides simple and efficient tools for predictive data analysis, and is reusable in various contexts, encouraging academic and commercial use. Scikit-learn's collection of algorithms and modules gave us the flexibility to construct, study, and improve our models in a time-efficient manner. For the implementation of GLVQ, we opted to use sklearn-glvq, a specialized module that extends the capabilities of scikit-learn to support Generalized Learning Vector Quantization. By employing sklearn-glvq, we were able to apply the robustness and versatility of scikit-learn to the nuanced requirements of GLVQ, further enriching our toolkit for investigating and illustrating the decision-making processes of these models.

The project's codebase is organized in a way that is both user-friendly and capable of scaling. It is divided into two primary sections: ”frontend” and ”backend”. The ”frontend” section contains the code that constructs the user interface (UI), developed using QML (Qt Meta-Object Language). QML is a declarative language that enables the creation of interactive and visually appealing UI elements.

The code in the 'frontend' and 'backend' areas is divided further into 'log' and 'ml' subcategories. The 'frontend/ml' segment contains the QML code tailored for ML.LOG, focusing on user interface components and features related to choosing models and displaying visuals. In contrast, the ”frontend/log” section houses the QML code for >_NEXT(LOG).

The ”backend” section contains the ”controller.py” file, which acts as the intermediary between the frontend UI and the underlying operations of ML.LOG. This file functions as a bridge, enabling the transfer of data and facilitating communication between the frontend UI and the backend logic.

The codebase is designed with a focus on user-friendliness and scalability, allowing for easy upkeep, growth, and modular development. This architectural design differentiates the roles of the UI and the backend operations, as well as the logic related to ML.LOG and >_NEXT(LOG), thereby ensuring a distinct separation of duties.

It is also worth noting that while designing the codebase for this project, we also took into account the potential need for expansion and adaptability. A key design goal was to ensure that integrating new models into the existing system would be as seamless and straightforward as possible. The code was architected in a modular fashion, with distinct sections for each model. This was accomplished by encapsulating the code specific to each model, allowing for their independent operation and maintenance. This way, when a new model needs to be added, the process would involve creating a new, self-contained module, mitigating potential disruption to the existing system.

4.2 Using ML.LOG

The ML.LOG program is a comprehensive tool designed to facilitate the application of machine learning techniques to adaptive business process logs. The program is structured around several key stages, each of which contributes to the overall process of training, evaluating, and applying machine learning models. For a visualization of the ML.LOG program, please refer to 7.2.

4.2.1 Home page

Upon initiating the ML.LOG program, the user's first view is the home page. The program requires an input in the form of a CSV file, which serves as the training and testing data. This CSV file can be generated using the >_NEXT(LOG) application or can be a general CSV file where the first $n-1$ columns represent the features and the n^{th} column corresponds to the labels, target, or classes column. The settings menu allows users to specify the CSV file to be parsed. Once the CSV file is successfully uploaded, additional functionalities become accessible, enabling users to proceed with their tasks.

4.2.2 Model page

The next stage involves model selection and parameter tuning. Users can browse through different models and select the one they wish to use. Once a model is selected, users can input the corresponding parameters. If users wish to explore a range of parameters to determine the best ones, they can input them as a list. For instance, for the Random Forest model, users can input "n_estimators" as [1, 2, 3, 4]. The program will then iterate through all the provided parameters and return the optimal one based on the F1 score. For string inputs, users should use the format ["gini", "entropy", "log_loss"], while for boolean values, 0 is used for False and 1 for True.

Additional input fields are provided for the KNN and LVQ models to specify the features to be plotted on the subsequent page. If only one feature is input, it will be plotted against itself. If two features are input, three plots will be generated: feature 1 against feature 1, feature 2 against feature 2, and feature 1 against feature 2. If three features are input, a 3D representation containing all three features will be generated. If these fields are left empty, the three most significant features will be automatically extracted using a Gradient Boosting Classifier.

The "Auto-ML" button allows users to activate an automatic training process that reads the contents of the "auto_ml.txt" file and executes the training process on all the models listed within the text file, disregarding any selections made in the GUI. The Results page will display the best model and its corresponding parameters, determined based on the F1 score. A slider is provided to choose the percentage of the dataset allocated for training/testing.

4.2.3 Results page

In the Results section, users can examine the outcomes of the (best) model. This section provides a comprehensive overview of the utilized parameters and various scores. Depending on the model, users can browse through images, view them in full screen, and save them for future reference. Users also have the option to save the model as a .pickle file, which retains important information including the timestamp of when the model was saved, the actual model itself, the percentage of the training split, the currently displayed image in the GUI, and the F1, precision, and recall scores.

4.2.4 Load & Test Pretrained Model page

The program also provides an option to upload a .pickle file containing a saved model. Users can upload a .csv file to evaluate the model's performance on a different dataset than the one it was originally trained on. Once the testing is complete, comprehensive statistics pertaining to the training process will be displayed.

5 Results and Analysis

This chapter presents the results of the machine learning models applied to the analysis of adaptive business process logs. The analysis was conducted on three different datasets, each representing a unique set of adaptive business processes. For each dataset, the results obtained from the analysis will be detailed.

The results section includes the performance metrics of the machine learning models, such as F1 score, recall, and accuracy. These metrics provide a quantitative measure of the models' effectiveness in identifying patterns or rules within the adaptive business process logs.

Following the presentation of results, the rules extracted from the best-performing model are discussed. These rules represent the correlations identified in the adaptive business process logs and provide valuable insights into the underlying business processes.

Finally, a comparison of the models is provided, evaluating their performance against each other to determine the most effective approach. This comparison is based on the F1 score and other metrics, providing a comprehensive assessment of the models' performance.

This chapter aims to provide a detailed and thorough analysis of the results, offering a clear understanding of the effectiveness of machine learning algorithms in analyzing adaptive business process logs. The findings from this chapter will form the basis for the discussion and conclusions drawn in the subsequent chapters.

Please note that finding the best parameters to use in GridSearchCV is specific to the dataset and is out of the scope of this thesis. The values suggested below are just starting points, and the optimal parameters may vary depending on the specific characteristics of your data.

This being said, the following parameters were used:

- **K-Nearest Neighbors (KNN):**
  - n_neighbors: [1, 3, 5, 7, 9]
  - weights: ["uniform", "distance"]
  - algorithm: ["auto", "ball_tree", "kd_tree", "brute"]
  - leaf_size: [10, 30, 50]
  - p: [1, 2]
- **Generalized Learning Vector Quantization (GLVQ):**
  - prototype_n_per_class: [1, 2, 3]
  - distance_type: ["squared-euclidean", "euclidean"]
  - activation_type: ["identity", "sigmoid", "soft+", "swish"]
  - solver_type: ["sgd", "wgd", "adam", "lbfgs", "bfgs"]
- **Decision Tree:**
  - criterion: ["gini", "entropy"]
  - max_depth: [5, 10, 15]
  - min_samples_split: [2, 5, 10]

  - min_samples_leaf: [1, 2, 3]
  - max_features: [”sqrt”, ”log2”]
  - max_leaf_nodes: [2, 4, 10]
  - splitter: [”best”, ”random”]

- **Random Forest:**
  - n_estimators: [10, 50, 100]
  - max_depth: [5, 10, 15]
  - max_leaf_nodes: [2, 4, 10]
  - min_samples_split: [2, 5, 10]
  - min_samples_leaf: [1, 2, 3]
  - max_features: [”sqrt”, ”log2”]
  - bootstrap: [1, 0]

- **Split:**
  - train: 80
  - test: 20

5.1 Dataset 1

This subsection is dedicated to discussing Dataset 1. As part of this analysis, a specific business process model is utilized. A visual representation of this model can be seen in Figure 1.

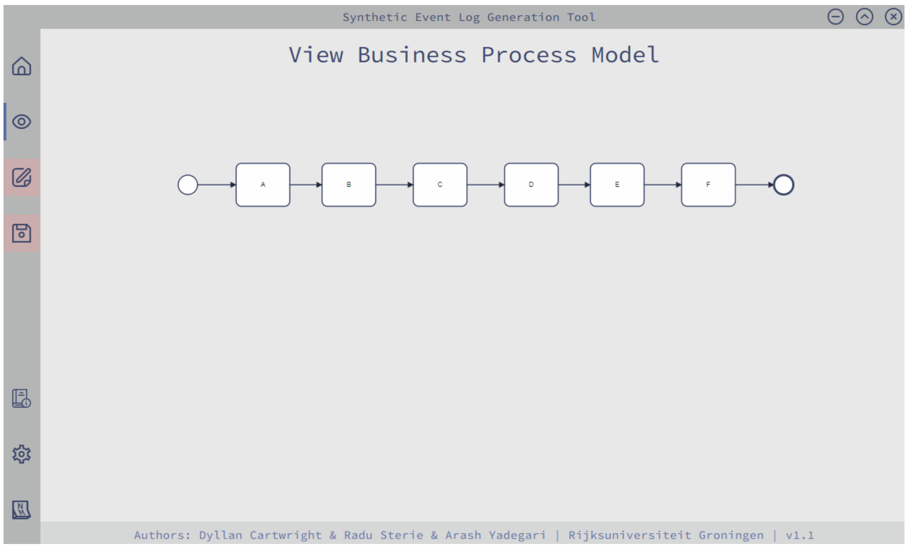

Synthetic Event Log Generation Tool
View Business Process Model
A
B
C
D
E
F
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.1

Figure 1: The business process model used for Dataset 1 visualized in >_NEXT(LOG).

This study employs a specific rule for the dataset. The rule is stated as follows:

```
if C#duration > 575 then insert K (#duration ?N(100 15));
```

The interpretation of this rule is that if the (cumulative) duration in event C exceeds 575, then an insert operation for event K with #duration is performed. The duration follows a normal distribution with a mean of 100 and a standard deviation of 15.

5.1.1 Results

In this subsubsection, the results derived from employing different configurations will be discussed in a detailed and elaborate manner.

Random Forest
In the case of the Random Forest algorithm, the optimal parameters were selected as follows:

- **Best hyperparameters:**
  - n_estimators: 50
  - max_depth: 5
  - max_leaf_nodes: 10
  - min_samples_split: 5
  - min_samples_leaf: 1
  - max_features: 'sqrt'
  - bootstrap: true

The corresponding accuracies for the train and test datasets are noted below:

```
Train Accuracy: 1.000
Test Accuracy: 1.000
```

The performance of the model for each class is reported in the following Table 1:

Table 1: Random Forest Classification Report for Dataset 1

	precision	recall	f1-score	support
class 0	1.00	1.00	1.00	276
class 1	1.00	1.00	1.00	24
macro avg	1.00	1.00	1.00	300
weighted avg	1.00	1.00	1.00	300

Decision Tree
The configuration for the Decision Tree model resulted in the optimal parameters as follows:

- **Best hyperparameters:**
  - criterion: ’gini’
  - max_depth: 5
  - min_samples_split: 2
  - min_samples_leaf: 3
  - max_features: ’sqrt’
  - max_leaf_nodes: 10
  - splitter: best

In terms of performance, the accuracies for the train and test datasets were:

```
Train Accuracy: 0.998
Test Accuracy: 0.990
```

The classification report for this model is provided in Table 2:

Table 2: Decision Tree Classification Report for Dataset 1

	precision	recall	f1-score	support
class 0	1.00	0.99	0.99	276
class 1	0.92	0.96	0.94	24
macro avg	0.96	0.98	0.97	300
weighted avg	0.99	0.99	0.99	300

KNN
In the case of the K-Nearest Neighbors model, the following set of hyperparameters yielded the optimal performance:

- **Best hyperparameters:**
  - n_neighbors: 1
  - weights: ”uniform”
  - algorithm: ”auto”
  - leaf_size: 10
  - p: 1

The KNN model demonstrated the following performance metrics:

```
Train Accuracy: 1.000
Test Accuracy: 0.987
```

Table 3: K-Nearest Neighbors Classification Report for Dataset 1

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	276
1.0	1.00	0.83	0.91	24
macro avg	0.99	0.92	0.95	300
weighted avg	0.99	0.99	0.99	300

The detailed classification results for the KNN model are depicted in Table 3:

GLVQ

For the Generalized Learning Vector Quantization model, the optimal configuration was achieved with the following set of hyperparameters:

- **Best hyperparameters:**
  - prototype_n_per_class: 3
  - distance_type: ”squared-euclidean”
  - activation_type: ”swish”
  - solver_type: ”sgd”

The performance of the GLVQ model was evaluated with the following metrics:

```
Train Accuracy: 0.976
Test Accuracy: 0.947
```

The comprehensive classification results for the GLVQ model are presented in Table 4:

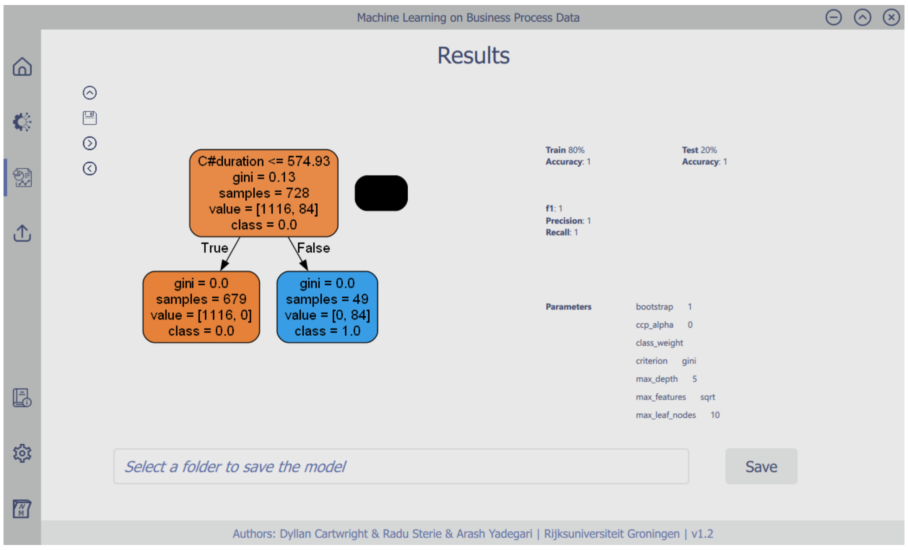
Table 4: Generalized Learning Vector Quantization Classification Report for Dataset 1

	precision	recall	f1-score	support
class 0	0.95	1.00	0.97	276
class 1	1.00	0.33	0.50	24
macro avg	0.97	0.67	0.74	300
weighted avg	0.95	0.95	0.93	300

5.1.2 Rule Extraction

Random Forest

Figure 2 illustrates the decision-making process for a single tree from the random forest.



Train 80%
Accuracy: 1

Test 20%
Accuracy: 1

f1: 1
Precision: 1
Recall: 1

Parameters		
	bootstrap	1
	ccp_alpha	0
	class_weight	
	criterion	gini
	max_depth	5
	max_features	sqrt
	max_leaf_nodes	10

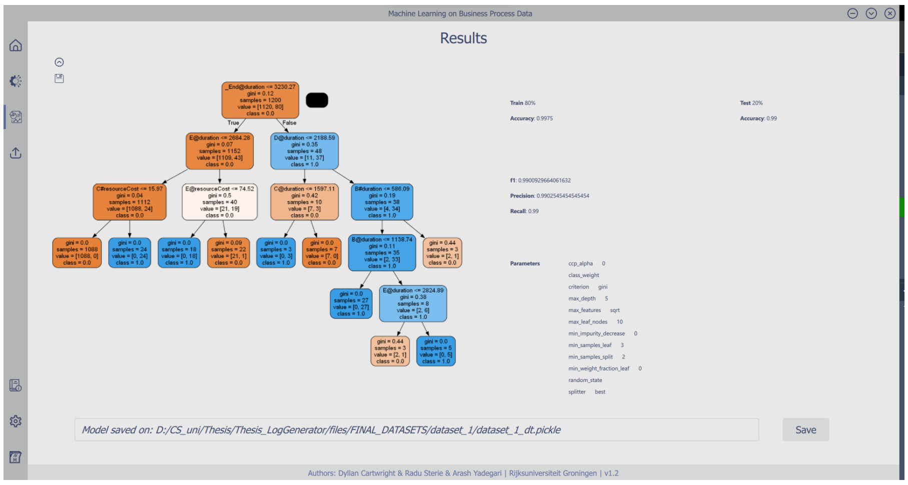
Select a folder to save the model

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 2: Visualization of decision making from a tree in the forest in ML.LOG for Dataset 1.

Decision Tree

Figures 3 and 4 illustrate the decision-making process within the decision tree model.



Train 80%
Accuracy: 0.9975

Test 20%
Accuracy: 0.99

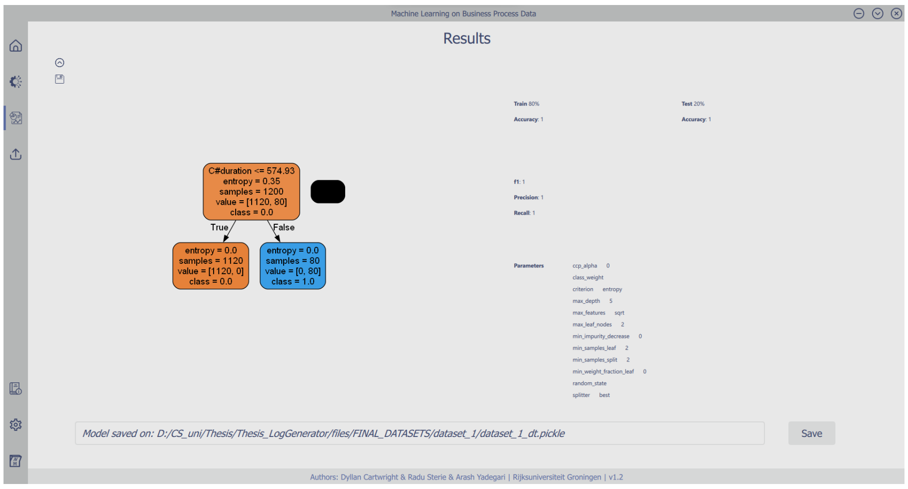
f1: 0.9900929664061632
Precision: 0.9902545454545454
Recall: 0.99

Parameters		
	ccp_alpha	0
	class_weight	
	criterion	gini
	max_depth	5
	max_features	sqrt
	max_leaf_nodes	10
	min_impurity_decrease	0
	min_samples_leaf	3
	min_samples_split	2
	min_weight_fraction_leaf	0
	random_state	
	splitter	best

Model saved on: D:/CS_uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_1/dataset_1_dt.pickle

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 3: Decision making visualization for a decision tree in ML.LOG for Dataset 1.



Train 80% **Test** 20%

Accuracy: 1 **Accuracy**: 1

f1: 1

Precision: 1

Recall: 1

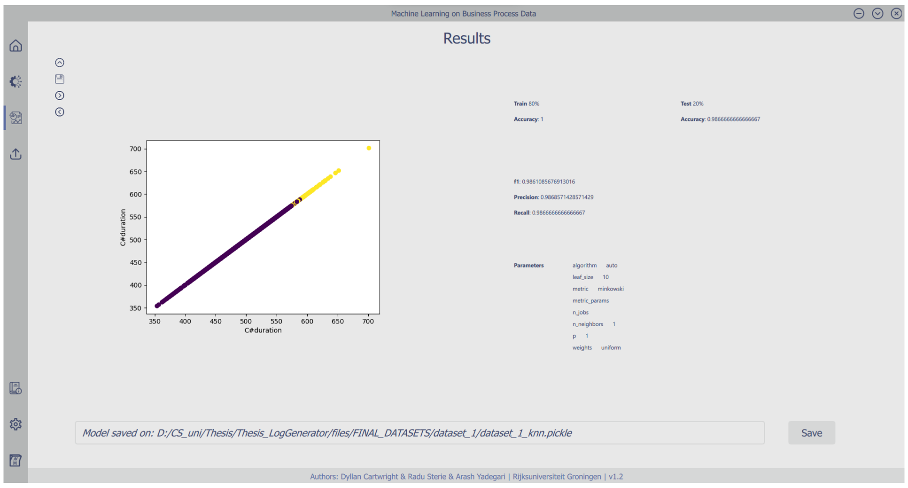
Parameters		
	ccp_alpha	0
	class_weight	
	criterion	entropy
	max_depth	5
	max_features	sqrt
	max_leaf_nodes	2
	min_impurity_decrease	0
	min_samples_leaf	2
	min_samples_split	2
	min_weight_fraction_leaf	0
	random_state	
	splitter	best

Model saved on: D:/CS_uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_1/dataset_1_dt.pickle

Figure 4: Visualization of decision making for a decision tree with $max\_leaf\_nodes = 2$ in ML.LOG for Dataset 1.

KNN

Figures 5 and 6 represent the labeling of the dataset by the K-Nearest Neighbors model.


Machine Learning on Business Process Data
Results
700
650
600
550
500
450
400
350
C#duration
350
400
450
500
550
600
650
700
C#duration
Save
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Train 80% **Test** 20%

Accuracy: 1 **Accuracy**: 0.9866666666666667

f1: 0.9861085676913016

Precision: 0.9868571428571429

Recall: 0.9866666666666667

Parameters		
	algorithm	auto
	leaf_size	10
	metric	minkowski
	metric_params	
	n_jobs	
	n_neighbors	1
	p	1
	weights	uniform

Model saved on: D:/CS_uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_1/dataset_1_knn.pickle

Figure 5: Visualization of labels assigned by KNN in ML.LOG for Dataset 1.

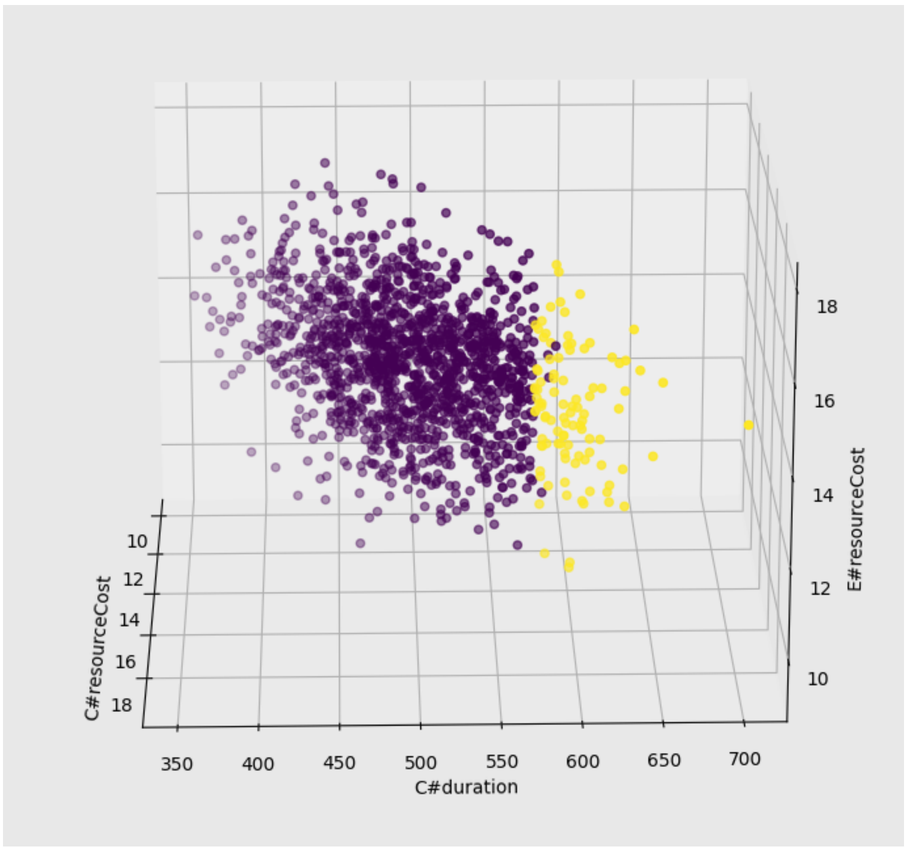
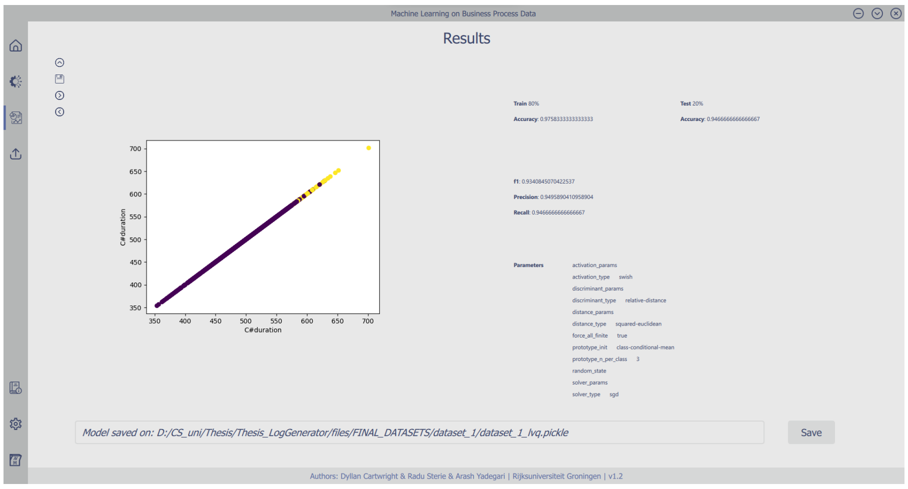


Figure 6: Visualization of labels assigned by KNN in ML.LOG for Dataset 1.

GLVQ

Figures 7 and 8 show the labeling of the dataset by the Generalized Learning Vector Quantization model.



Train 80%

Accuracy: 0.9758333333333333

Test 20%

Accuracy: 0.9466666666666667

f1: 0.9340845070422537

Precision: 0.9495890410958904

Recall: 0.9466666666666667

Parameters		
	activation_params	
	activation_type	swish
	discriminant_params	
	discriminant_type	relative-distance
	distance_params	
	distance_type	squared-euclidean
	force_all_finite	true
	prototype_init	class-conditional-mean
	prototype_n_per_class	3
	random_state	
	solver_params	
	solver_type	sgd

Model saved on: D:/CS_uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_1/dataset_1_lvq.pickle

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 7: 2D visualization of labels assigned by GLVQ in ML.LOG for Dataset 1.

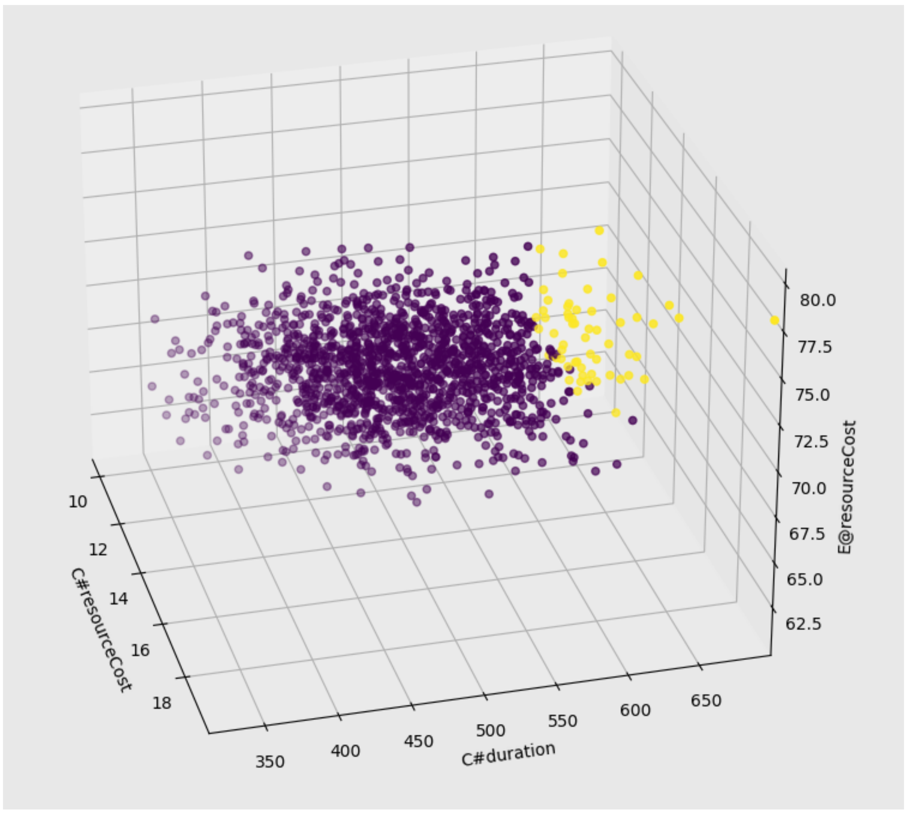


Figure 8: 3D visualization of labels assigned by GLVQ in ML.LOG for Dataset 1.

5.2 Dataset 2

This subsection is devoted to the exploration of Dataset 2. In this analysis, a distinct business process model is employed. A graphical depiction of this model is presented in Figure 9.

A unique rule is applied to this dataset. The rule is articulated as follows:

```
if A#duration > 575 then skip C;
```

The interpretation of this rule is that if the accumulated duration in event A surpasses 575, a skip operation for event C is executed.

5.2.1 Results

This subsubsection will provide a comprehensive discussion on the results obtained from different configurations.

Random Forest

The optimal parameters for the Random Forest algorithm were chosen as follows:

- **Best hyperparameters:**
  - n_estimators: 50
  - max_depth: 5

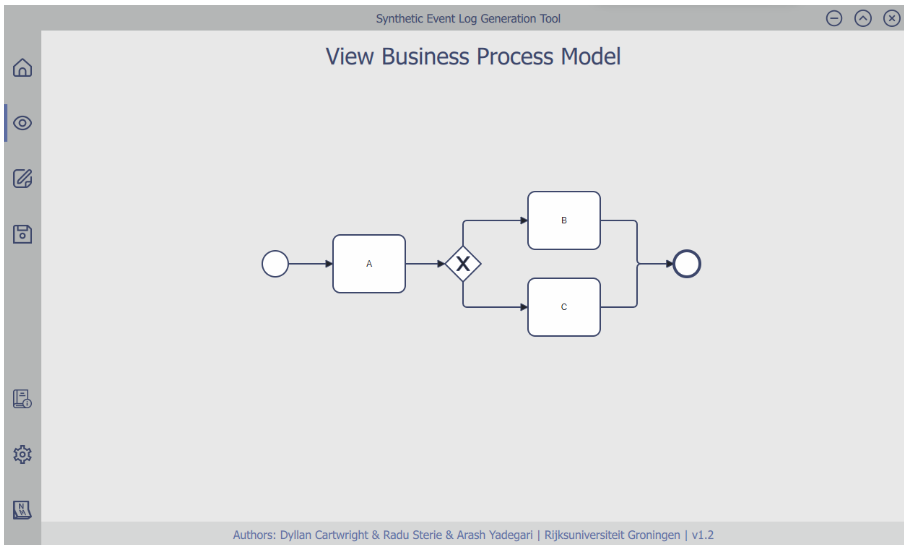

Synthetic Event Log Generation Tool
View Business Process Model
A
B
C
X
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 9: The business process model applied to Dataset 2 visualized in >_NEXT(LOG).

- max_leaf nodes: 2
- min_samples_split: 10
- min_samples_leaf: 1
- max_features: "sqrt"
- bootstrap: 1

The corresponding accuracies for the training and testing datasets are noted below:

```
Train Accuracy: 0.998
Test Accuracy: 1.00
```

The performance of the model for each class is reported in the following Table 5:

Table 5: Random Forest Classification Report for Dataset 2

	precision	recall	f1-score	support
class 0	1.0	1.0	1.0	293
class 1	1.0	1.0	1.0	7
macro avg	1.0	1.0	1.0	300
weighted avg	1.0	1.0	1.0	300

Decision Tree

The configuration for the Decision Tree model resulted in the optimal parameters as follows:

- **Best hyperparameters:**
  - criterion: 'gini'
  - max_depth: 5

  - min_samples_split: 5
  - min_samples_leaf: 1
  - max_features: ’sqrt’
  - max_leaf_nodes: 4
  - splitter: ’best’

In terms of performance, the accuracies for the training and testing datasets were:

```
Train Accuracy: 1.000
Test Accuracy: 1.000
```

The classification report for this model is provided in Table 6:

Table 6: Decision Tree Classification Report for Dataset 2

	precision	recall	f1-score	support
class 0	1.0	1.0	1.0	293
class 1	1.0	1.0	1.0	7
macro avg	1.0	1.0	1.0	300
weighted avg	1.0	1.0	1.0	300

KNN

For the K-Nearest Neighbors model, the following set of hyperparameters provided the best performance:

- **Best hyperparameters:**
  - n_neighbors: 1
  - weights: ”uniform”
  - algorithm: ”auto”
  - leaf_size: 10
  - p: 1

The KNN model demonstrated the following performance metrics:

```
Train Accuracy: 1.000
Test Accuracy: 1.000
```

The detailed classification results for the KNN model are depicted in Table 7:

GLVQ

For the Generalized Learning Vector Quantization model, the best configuration was achieved with the following set of hyperparameters:

- **Best hyperparameters:**

Table 7: K-Nearest Neighbors Classification Report for Dataset 2

	precision	recall	f1-score	support
class 0	1.0	1.0	1.0	293
class 1	1.0	1.0	1.0	7
macro avg	1.0	1.0	1.0	300
weighted avg	1.0	1.0	1.0	300

- prototype_n_per_class: 2
- distance_type: ”squared-euclidean”
- activation_type: ”identity”
- solver_type: ”sgd”

The performance of the GLVQ model was evaluated with the following metrics:

```
Train Accuracy: 1.000
Test Accuracy: 1.000
```

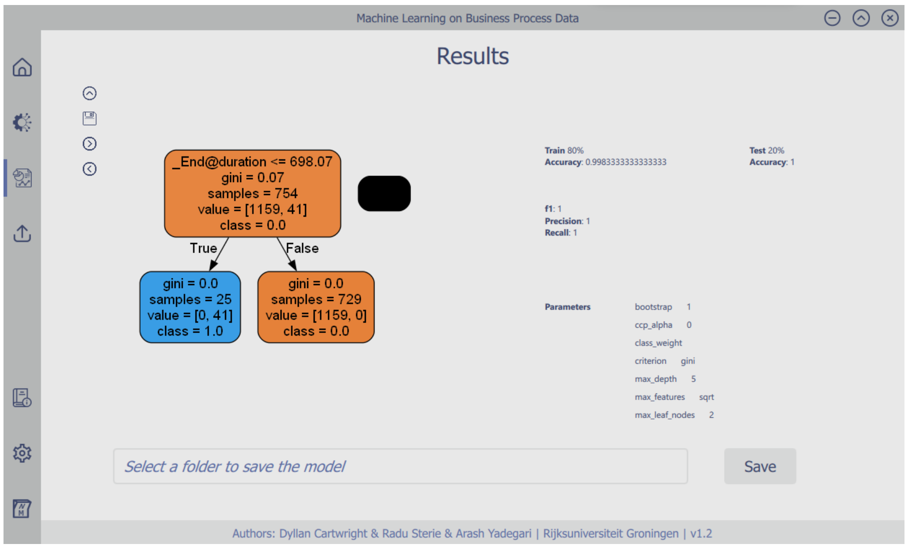
The comprehensive classification results for the GLVQ model are presented in Table 8:

Table 8: GLVQ Classification Report for Dataset 2

	precision	recall	f1-score	support
0.0	1.0	1.0	1.0	293
1.0	1.0	1.0	1.0	7
macro avg	1.0	1.0	1.0	300
weighted avg	1.0	1.0	1.0	300

5.2.2 Rule Extraction

Random Forest
Figure 10 illustrates the decision-making process for a single tree from the random forest.



Machine Learning on Business Process Data

Results

Train 80%
Accuracy: 0.9983333333333333

Test 20%
Accuracy: 1

f1: 1
Precision: 1
Recall: 1

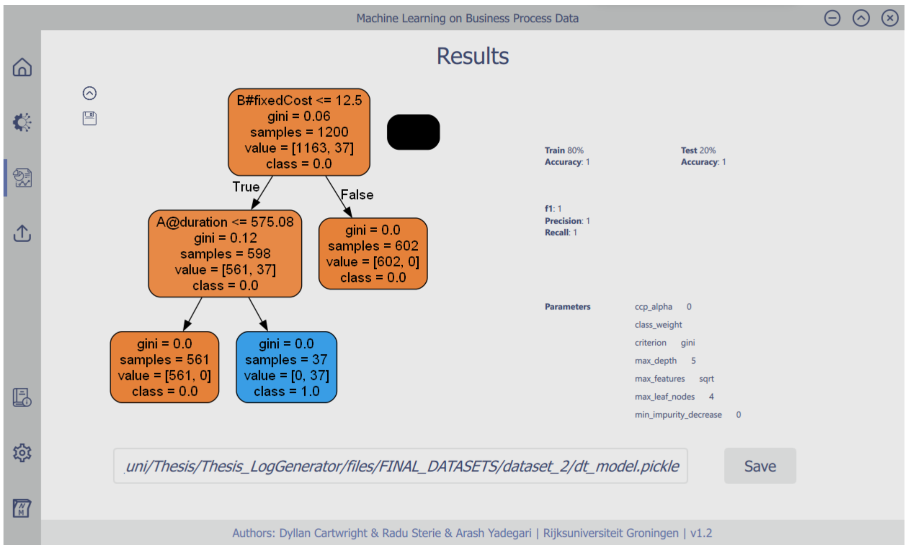
Parameters		
	bootstrap	1
	ccp_alpha	0
	class_weight	
	criterion	gini
	max_depth	5
	max_features	sqrt
	max_leaf_nodes	2

Select a folder to save the model

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 10: Decision making visualization for a decision tree in the forest ML.LOG for Dataset 2

Decision Tree
Figures 11 and 12 illustrate the decision-making process within the decision tree model.



Machine Learning on Business Process Data

Results

Train 80%
Accuracy: 1

Test 20%
Accuracy: 1

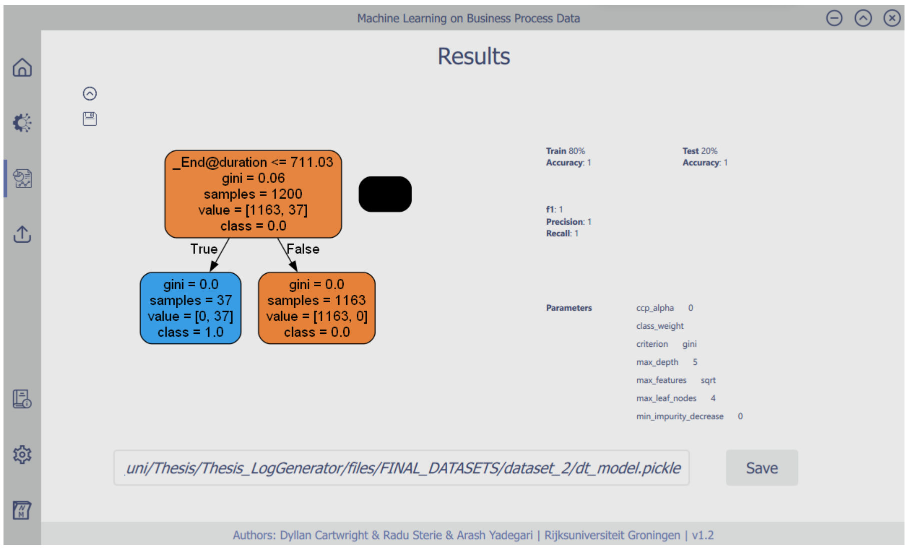
f1: 1
Precision: 1
Recall: 1

Parameters		
	ccp_alpha	0
	class_weight	
	criterion	gini
	max_depth	5
	max_features	sqrt
	max_leaf_nodes	4
	min_impurity_decrease	0

uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_2/dt_model.pickle

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 11: Decision making visualization for a decision tree in ML.LOG for Dataset 2.



Train 80%
Accuracy: 1

Test 20%
Accuracy: 1

f1: 1
Precision: 1
Recall: 1

Parameters

ccp_alpha	0
class_weight	
criterion	gini
max_depth	5
max_features	sqrt
max_leaf_nodes	4
min_impurity_decrease	0

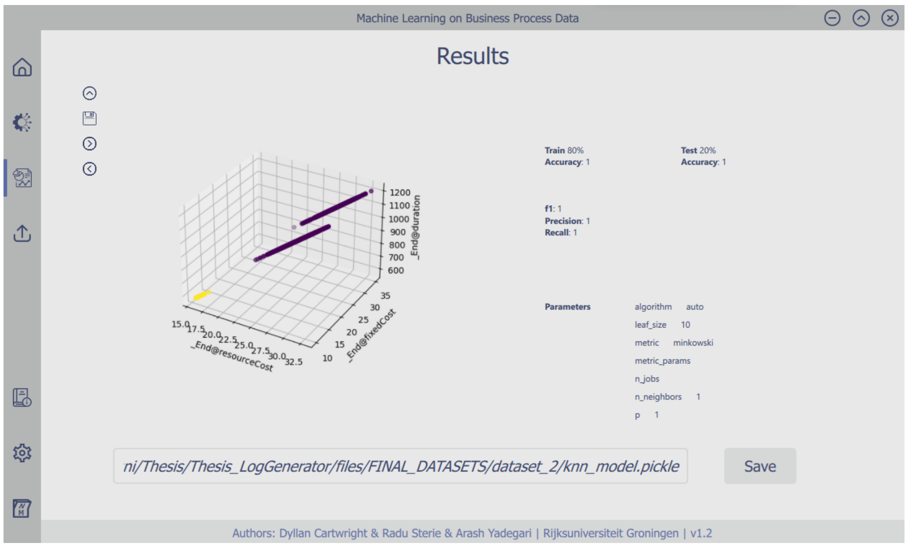
uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_2/dt_model.pickle

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 12: Decision making visualization for a decision tree in ML.LOG for Dataset 2.

KNN

Figure 13 represent the labeling of the dataset by the K-Nearest Neighbors model.



Train 80%
Accuracy: 1

Test 20%
Accuracy: 1

f1: 1
Precision: 1
Recall: 1

Parameters

algorithm	auto
leaf_size	10
metric	minkowski
metric_params	
n_jobs	
n_neighbors	1
p	1

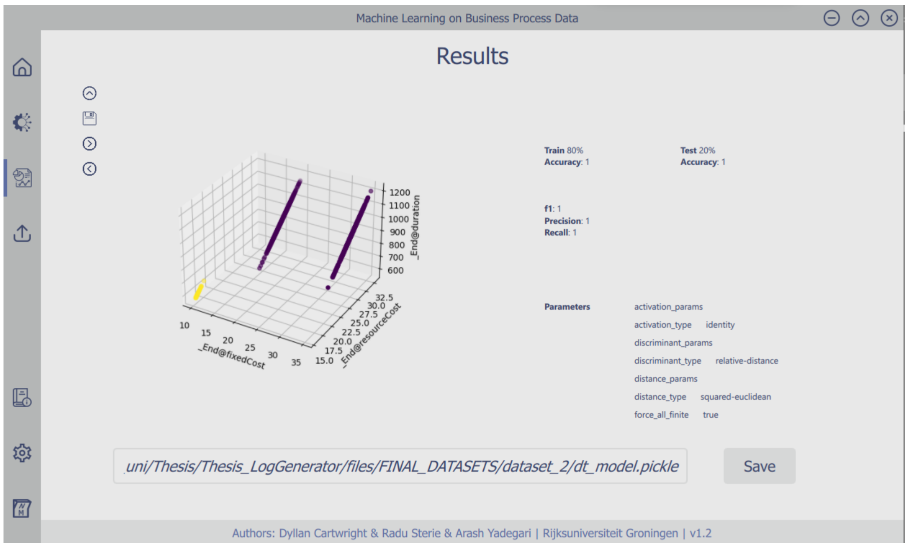
ni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_2/knn_model.pickle

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 13: 3D visualization of labels assigned by KNN in ML.LOG for Dataset 2.

GLVQ

Figure 14 shows the labeling of the dataset by the Generalized Learning Vector Quantization model.



Train 80%
Accuracy: 1

Test 20%
Accuracy: 1

f1: 1
Precision: 1
Recall: 1

Parameters		
	activation_params	
	activation_type	identity
	discriminant_params	
	discriminant_type	relative-distance
	distance_params	
	distance_type	squared-euclidean
	force_all_finite	true

uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_2/dt_model.pickle

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 14: 3D visualization of labels assigned by GLVQ in ML.LOG for Dataset 2.

5.3 Dataset 3

This subsection is allocated to the examination of Dataset 3. In this part of the analysis, a particular business process model is adopted. A pictorial representation of this model is illustrated in Figure 15.

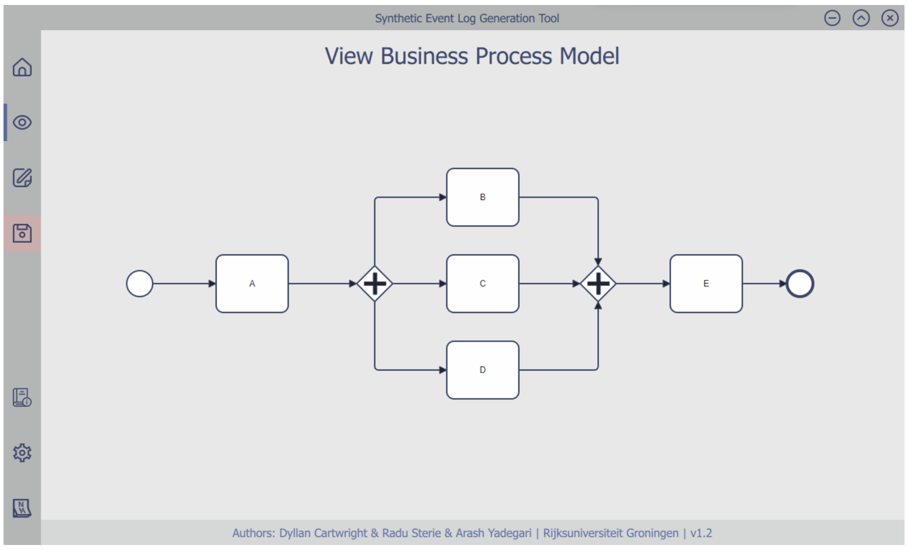

Synthetic Event Log Generation Tool
View Business Process Model
A
B
C
D
E
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 15: The business process model applied to Dataset 3 visualized in >_NEXT(LOG).

A distinct rule is implemented for this dataset. The rule is expressed as follows:

```
% if A#duration < 425 then C–B–D;
```

The interpretation of this rule is that if the total duration in event A is below 425, then C, B and D will be executed in series. Note that for this rule to make sense, C, B and D have to be in parallel.

5.3.1 Results

This subsubsection presents a detailed discussion on the results obtained from various configurations.

Random Forest

The optimal parameters for the Random Forest algorithm were selected as follows:

- **Best hyperparameters:**
  - n_estimators: 100
  - max_depth: 5
  - max_leaf_nodes: 2
  - min_samples_split: 2
  - min_samples_leaf: 1
  - max_features: ”sqrt”
  - bootstrap: 1

The corresponding accuracies for the training and testing datasets are noted below:

```
Train Accuracy: 1.000
Test Accuracy: 1.000
```

The performance of the model for each class is reported in the following table:

Table 9: Random Forest Classification Report for Dataset 3

	precision	recall	f1-score	support
class 0	1.0	1.0	1.0	269
class 1	1.0	1.0	1.0	31
macro avg	1.0	1.0	1.0	300
weighted avg	1.0	1.0	1.0	300

Decision Tree

The configuration for the Decision Tree model resulted in the optimal parameters as follows:

- **Best hyperparameters:**
  - criterion: ’gini’
  - max_depth: 5
  - min_samples_split: 5
  - min_samples_leaf: 1
  - max_features: ’sqrt’
  - max_leaf_nodes: 4
  - splitter: ”best”

In terms of performance, the accuracies for the training and testing datasets were:

Table 10: Decision Tree Classification Report for Dataset 3

	precision	recall	f1-score	support
class 0	1.0	1.0	1.0	269
class 1	1.0	1.0	1.0	31
macro avg	1.0	1.0	1.0	300
weighted avg	1.0	1.0	1.0	300

```
Train Accuracy: 1.000
Test Accuracy: 1.000
```

The classification report for this model is provided in Table 10:

KNN

For the K-Nearest Neighbors model, the following set of hyperparameters provided the best performance:

- **Best hyperparameters:**
  - n_neighbors: 3
  - weights: ”distance”
  - algorithm: ”auto”
  - leaf_size: 10
  - p: 1

The KNN model demonstrated the following performance metrics:

```
Train Accuracy: 1.000
Test Accuracy: 0.983
```

The detailed classification results for the KNN model are depicted in Table 11:

Table 11: K-Nearest Neighbors Classification Report for Dataset 3

	precision	recall	f1-score	support
0.0	0.98	1.00	0.99	269
1.0	1.00	0.84	0.91	31
macro avg	0.99	0.92	0.95	300
weighted avg	0.98	0.98	0.98	300

GLVQ

For the Generalized Learning Vector Quantization model, the best configuration was achieved with the following set of hyperparameters:

- **Best hyperparameters:**

- prototype_n_per_class: 3
- distance_type: ”squared-euclidean”
- activation_type: ”swish”
- solver_type: ”sgd”

The performance of the GLVQ model was evaluated with the following metrics:

```
Train Accuracy: 0.980
Test Accuracy: 0.963
```

The comprehensive classification results for the GLVQ model are presented in Table 12:

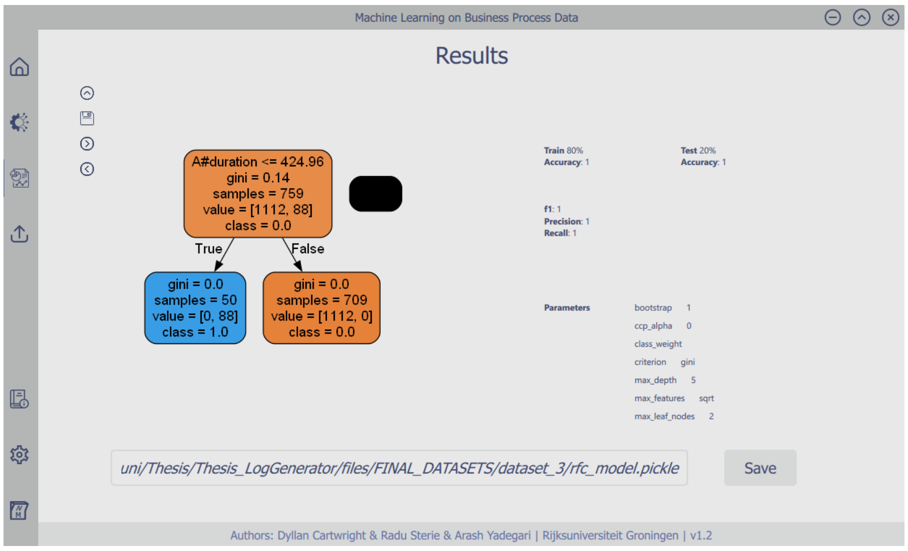
Table 12: GLVQ Classification Report for Dataset 3

	precision	recall	f1-score	support
0.0	0.96	1.00	0.98	269
1.0	1.00	0.65	0.78	31
macro avg	0.98	0.82	0.88	300
weighted avg	0.96	0.96	0.96	300

5.3.2 Rule Extraction

Random Forest

Figure 16 illustrates the decision-making process for a single tree from the random forest.



Train 80%
Accuracy: 1

Test 20%
Accuracy: 1

f1: 1
Precision: 1
Recall: 1

Parameters

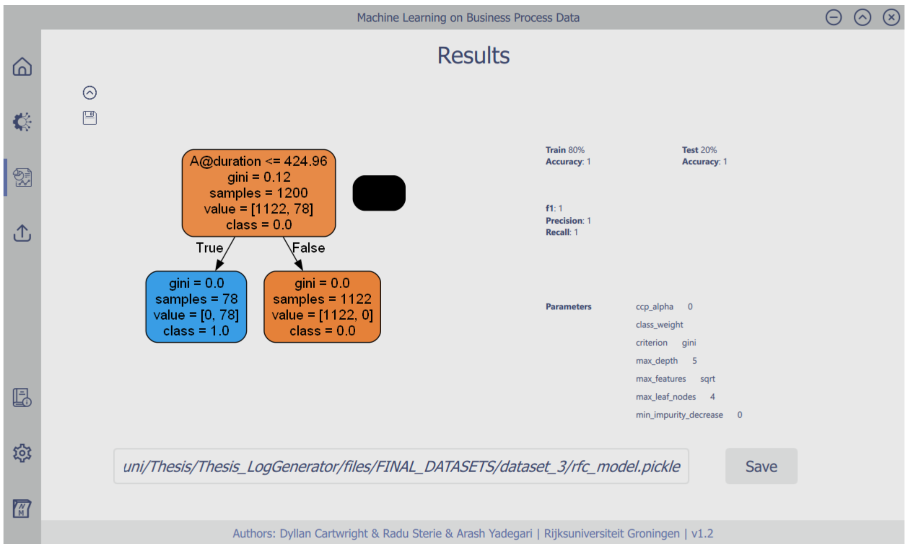
bootstrap	1
ccp_alpha	0
class_weight	
criterion	gini
max_depth	5
max_features	sqrt
max_leaf_nodes	2

uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/rfc_model.pickle

Figure 16: Decision making visualization for a decision tree in the forest ML.LOG

Decision Tree

Figure 17 illustrates the decision-making process within the decision tree model.



Train 80%
Accuracy: 1

Test 20%
Accuracy: 1

f1: 1
Precision: 1
Recall: 1

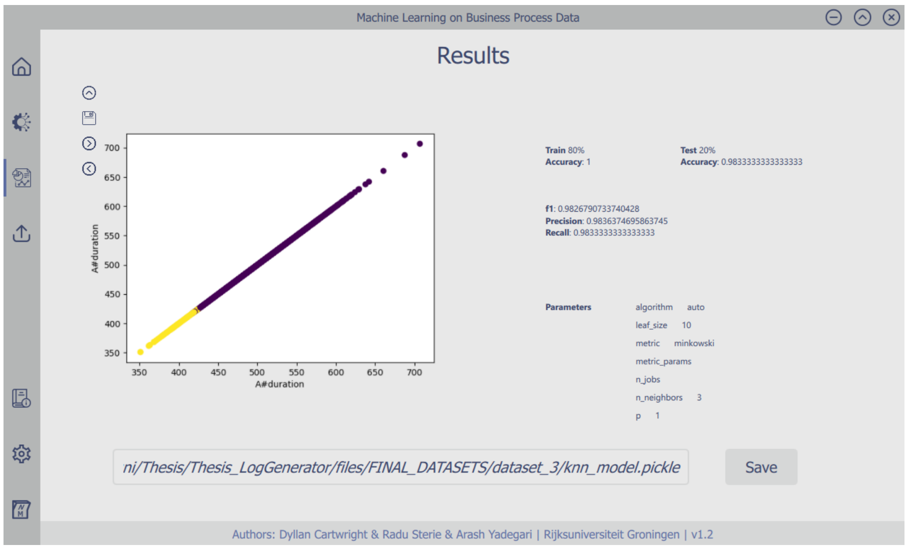
Parameters		
	ccp_alpha	0
	class_weight	
	criterion	gini
	max_depth	5
	max_features	sqrt
	max_leaf_nodes	4
	min_impurity_decrease	0

uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/rfc_model.pickle

Figure 17: Decision making visualization for a decision tree in ML.LOG.

KNN

Figure 18 represent the labeling of the dataset by the K-Nearest Neighbors model.


Machine Learning on Business Process Data
Results
700
650
600
550
500
450
400
350
A#duration
350
400
450
500
550
600
650
700
A#duration
Save
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Train 80%
Accuracy: 1

Test 20%
Accuracy: 0.9833333333333333

f1: 0.9826790733740428
Precision: 0.9836374695863745
Recall: 0.9833333333333333

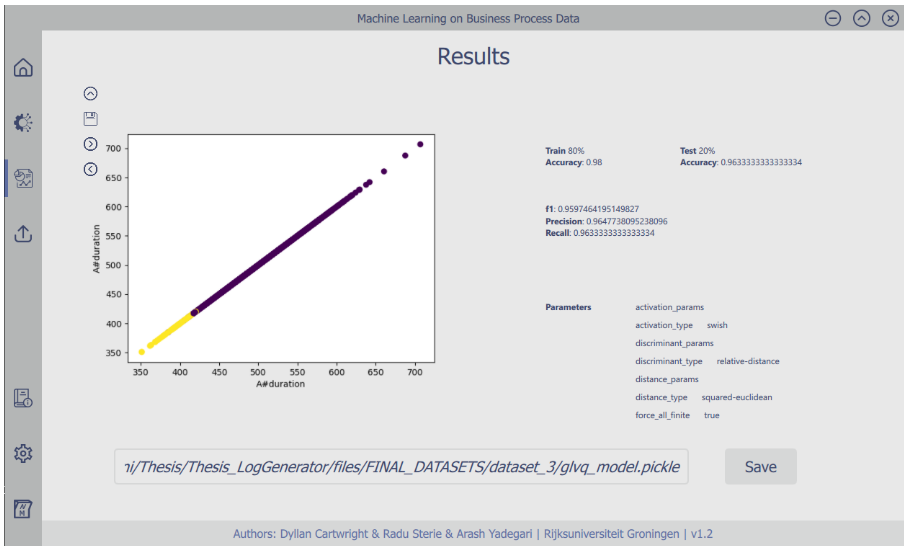
Parameters		
	algorithm	auto
	leaf_size	10
	metric	minkowski
	metric_params	
	n_jobs	
	n_neighbors	3
	p	1

ni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/knn_model.pickle

Figure 18: 3D visualization of labels assigned by KNN in ML.LOG.

GLVQ

Figure 19 shows the labeling of the dataset by the Generalized Learning Vector Quantization model.


Machine Learning on Business Process Data
Results
A#duration
A#duration
Save
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Train 80%
Accuracy: 0.98

Test 20%
Accuracy: 0.9633333333333334

f1: 0.9597464195149827
Precision: 0.9647738095238096
Recall: 0.9633333333333334

Parameters		
	activation_params	
	activation_type	swish
	discriminant_params	
	discriminant_type	relative-distance
	distance_params	
	distance_type	squared-euclidean
	force_all_finite	true

ıi/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/glvq_model.pickle

Figure 19: 3D visualization of labels assigned by GLVQ in ML.LOG.

5.4 Model Comparison

In this section, we delve deeper into the comparison of the machine learning models applied to the three datasets. The models' performance is evaluated based on key metrics such as F1 score, recall, and accuracy, which provide a quantitative measure of their effectiveness.

The Random Forest algorithm, with varying hyperparameters, demonstrated optimal performance across the datasets. For the first dataset, the model achieved perfect accuracy with hyperparameters such as n estimators: 50, max depth: 5, max leaf nodes: 10, min samples split: 5, min samples leaf: 1, max features: 'sqrt', and bootstrap: true. This indicates a perfect fit of the model to the data, suggesting that the Random Forest algorithm was able to capture the underlying patterns in the data effectively.

In contrast, for the third dataset, the optimal parameters for the Random Forest algorithm were slightly different, with n estimators: 100, max depth: 5, max leaf nodes: 2, min samples split: 2, min samples leaf: 1, max features: 'sqrt', bootstrap: 1. This variation in optimal hyperparameters across datasets highlights the adaptability of the Random Forest algorithm to different data characteristics. It suggests that the algorithm can adjust its complexity based on the structure of the data, which is a desirable property in machine learning models.

Despite these differences in hyperparameters, the Random Forest algorithm consistently demonstrated high performance across all datasets. This underscores its robustness and versatility in analyzing adaptive business process logs, making it a reliable choice for this type of analysis.

However, it's important to note that while the Random Forest algorithm performed well in these instances, it may not always be the best choice for every dataset or problem. Other machine learning models could potentially yield better results depending on the specific characteristics of the data and the problem at hand. Therefore, it's crucial to consider a variety of models when conducting machine learning analysis.

Future work should consider exploring other machine learning models and evaluation metrics to provide a more comprehensive comparison and understanding of their performance in the context of adaptive business process log analysis. This could include models like Support Vector Machines, or Neural Networks, each of which has its own strengths and may perform differently on these datasets.

Additionally, other evaluation metrics such as precision, ROC AUC, or log loss could be considered to provide different perspectives on model performance.

6 Discussion

The analysis of adaptive business process logs using machine learning models has yielded significant insights. The application of four different models, namely GLVQ, KNN, Decision Trees, and Random Forest, has allowed for a comprehensive exploration of the data. The performance of these models was evaluated using various metrics, including F1 score, recall, and accuracy, providing a quantitative measure of their effectiveness.

6.1 Interpretation of Results

The results obtained from the analysis of the adaptive business process logs were detailed and varied across the different models. For instance, the Random Forest algorithm demonstrated optimal performance with hyperparameters such as n estimators: 50, max depth: 5, max leaf nodes: 10, min samples split: 5, min samples leaf: 1, max features: 'sqrt', and bootstrap: true. The corresponding accuracies for the train and test datasets were both 1.000, indicating a perfect fit of the model to the data.

Upon careful analysis of Figure 16, it becomes apparent that the "extracted" rule is indeed correct, aligning well with the expected behavior. Likewise, Figure 17 illustrates another precise extracted rule. However, it is important to highlight that during this process, additional trees with a greater number of nodes were discovered 3. While these trees also exhibited (almost) perfect accuracy, the extra nodes lacked significance and instead represented random correlations that were identified.

Extrapolating rules from KNN is a challenging task; however, ML.LOG provides the capability to visualize the labels assigned by KNN. Upon examining 18, it is observed that KNN recognizes the importance of C#duration, revealing an invisible yet distinct boundary at approximately 575. Consequently, this enables us to "extract" the presence of a rule associated with this Key Performance Indicator (KPI) and the corresponding threshold value.

Similar to KNN, GLVQ exhibits a comparable pattern, as demonstrated in 19. Despite this being a relatively simple setup and dataset, it is noteworthy that all techniques showcased remarkably high accuracy.

In the context of the feature plotting for KNN and LVQ, it is worth mentioning that the features to be shown in the plots were not selected manually. Instead, these features were automatically chosen through the application of a Gradient Boosting Classifier. The process resulted in the identification of the determining feature for the labels in both Dataset 1 and Dataset 3. However, in the case of Dataset 2, different features were selected.

This is due to the fact that a nearly 1:1 relationship was observed between the "End@duration" and the rule itself. This correlation arises due to the design of the dataset, where the duration of each event was generated following a normal distribution with a mean of 500 and a standard deviation of 50. Consequently, the majority of instances will demonstrate a total duration of approximately 1000 in the absence of event C being skipped. If event C is skipped, the total duration tends towards 575, though it may extend up to around 700, which is equivalent to four standard deviations above the mean. This can also be seen in the case of the Random Forest model, specifically Figure 10.

This results in a high degree of accuracy, suggesting a causal relationship originating from the rule, as opposed to the rule itself being the principal determinant. This insight necessitates further exploration to more accurately distinguish between correlation and causation in the observed relationship.

6.2 Contributions and Limitations

This study contributes to the field of adaptive business process analysis and machine learning by providing a tool that can help organizations better understand and optimize their business processes. The research has demonstrated the potential of machine learning models in identifying patterns or rules within adaptive business process logs, thereby providing valuable insights into the underlying business processes.

However, there are several limitations to this study. The research is based on the analysis of adaptive business process logs generated by the >_NEXT(LOG)program, and the findings may not be generalizable to other types of business process logs or to logs generated by different programs. While the study compared the performance of four different machine learning models, there are many other models that could potentially be used for this task. The results of this study may not be applicable to these other models. The evaluation of the models' performance is based on specific metrics such as F1 score, recall, and accuracy. These metrics may not capture all aspects of a model's performance, and other metrics could potentially yield different results.

7 Conclusion

7.1 Summary of the Research

The research conducted in this study aimed to explore the potential of machine learning techniques in identifying correlations within adaptive business process logs. The study employed a novel methodology, utilizing algorithms such as Generalized Learning Vector Quantization (GLVQ), K-Nearest Neighbors (KNN), Decision Trees, and Random Forests to analyze the data. The results were evaluated based on performance metrics such as F1 score, recall, and accuracy, providing a quantitative measure of the effectiveness of the models 5.

The analysis was conducted on three distinct datasets, each representing a unique set of adaptive business processes. The rules extracted from the best-performing model provided valuable insights into the underlying business processes, revealing correlations within the adaptive business process logs 5.

In conclusion, the research presented in this study has demonstrated the potential of machine learning techniques in the analysis of adaptive business process logs. Despite the limitations, the findings are anticipated to spur additional research and advancements in the field of adaptive business process analysis. Future work should consider the limitations identified in this study and explore the use of other machine learning models and evaluation metrics.

7.2 Auxiliary Information

Division of Tasks

I was the only student who was formally responsible for this project, so there were no division of tasks. Although, do note, Dyllan and I shared code/ideas throughout, and eventually combined >_NEXT(LOG) and ML.LOG into one program.

Deliverables

- The final BSc thesis.
- Software source code (and/or a compiled program).
- Final presentation.

All will be emailed to Arash, but nonetheless, can also be found here.

Grading

- Scientific quality of research and technical contribution: 40%.
- Project management and interpersonal skills: 20%.
- Final presentation: 20%.
- Report/Thesis: 20%.

Ethical Declaration

I hereby declare that this thesis presented herein is the result of my original research work. I assert that this thesis has not been submitted, either wholly or partially, for any other academic degree or qualification at any other university or institution.
All sources, including published or unpublished works, have been duly acknowledged and referenced, except where explicitly indicated. I affirm that the content of this thesis is the product of my independent efforts (unless stated) and reflects my understanding and interpretation of the subject matter.

Radu Andrei, s4151615, August 10, 2023

university of groningen

faculty of science and engineering

Radu Sterie

Bibliography

[1] Z. Bozorgi, I. Teinemaa, M. Dumas, M. Rosa, and A. Polyvyanyy, “Process mining meets causal machine learning: Discovering causal rules from event logs,” in *ICPM*, p. 28, 2020.

[2] J. Isabona, A. Imoize, and Y. Kim, “Machine learning-based boosted regression ensemble combined with hyperparameter tuning for optimal adaptive learning,” *Sensors*, vol. 22, no. 10, p. 3776, 2022.

[3] M. Camargo, M. Dumas, and O. G. Rojas, “Learning accurate business process simulation models from event logs via automated process discovery and deep learning,” in *BPM*, p. 4, 2021.

[4] A. Y. Ghahderijani and D. Karastoyanova, “Applying decision trees as a mean for correlation-identification and learning from adapted business process cases,” *University of Groningen*, 2023. *Unpublished as of yet.

[5] A. Y. Ghahderijani and D. Karastoyanova, “Synthetic event log generation for kpi-based process adaptations using simulation,” *University of Groningen*, 2023. *Unpublished as of yet.

[6] R. Conforti, M. Rosa, and A. T. Hofstede, “Filtering out infrequent behavior from business process event logs,” *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[7] A. Kalenkova, W. M. van der Aalst, I. Lomazova, and V. Rubin, “Process mining using bpmn: relating event logs and process models,” *Software & Systems Modeling*, 2016.

[8] G. Elkoumy, A. Pankova, and M. Dumas, “Mine me but don’t single me out: Differentially private event logs for process mining,” in *IEEE International Conference on Process Mining (ICPM)*, 2021.

[9] T. Nolle, S. Luettgen, A. Seeliger, and M. Muehlhaeuser, “Analyzing business process anomalies using autoencoders,” *Machine Learning*, vol. 107, pp. 1875–1893, 2018.

[10] F. Taymouri, M. Rosa, and J. Carmona, “Business process variant analysis based on mutual fingerprints of event logs,” in *International Conference on Business Process Management*, 2019.

[11] D. Cartwright, “A tool for log generation of adaptive business processes,” *University of Groningen*, 2023. *Unpublished as of yet.

[12] S. Islam and S. H. Amin, “Prediction of probable backorder scenarios in the supply chain using distributed random forest and gradient boosting machine learning techniques,” *Journal of Big Data*, vol. 7, no. 1, pp. 1–19, 2020.

[13] N. Ahmed, R. Amin, H. Aldabbas, D. Koundal, B. Alouffi, and T. Shah, “Machine learning techniques for spam detection in email and iot platforms: Analysis and research challenges,” *Security and Communication Networks*, vol. 2022, p. 1862888, 2022.

[14] J. Palša, N. Ádám, J. Hurtuk, E. Chovancová, B. Madoš, M. Chovanec, and S. Kocan, “Mlmd—a malware-detecting antivirus tool based on the xgboost machine learning algorithm,” *Applied Sciences*, vol. 12, no. 13, p. 6672, 2022.

[15] D. Krstajic, L. Buturovic, D. Leahy, and S. Thomas, “Cross-validation pitfalls when selecting and assessing regression and classification models,” *Journal of Cheminformatics*, vol. 6, no. 10, 2014.

[16] L. Ali, F. Alnajjar, H. Al Jassmi, M. Gochoo, W. Khan, and M. A. Serhani, “Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures,” *Sensors*, vol. 21, no. 5, p. 1688, 2021.

Appendices

The purpose of this section is to offer a more visual look at the ML.LOG program.

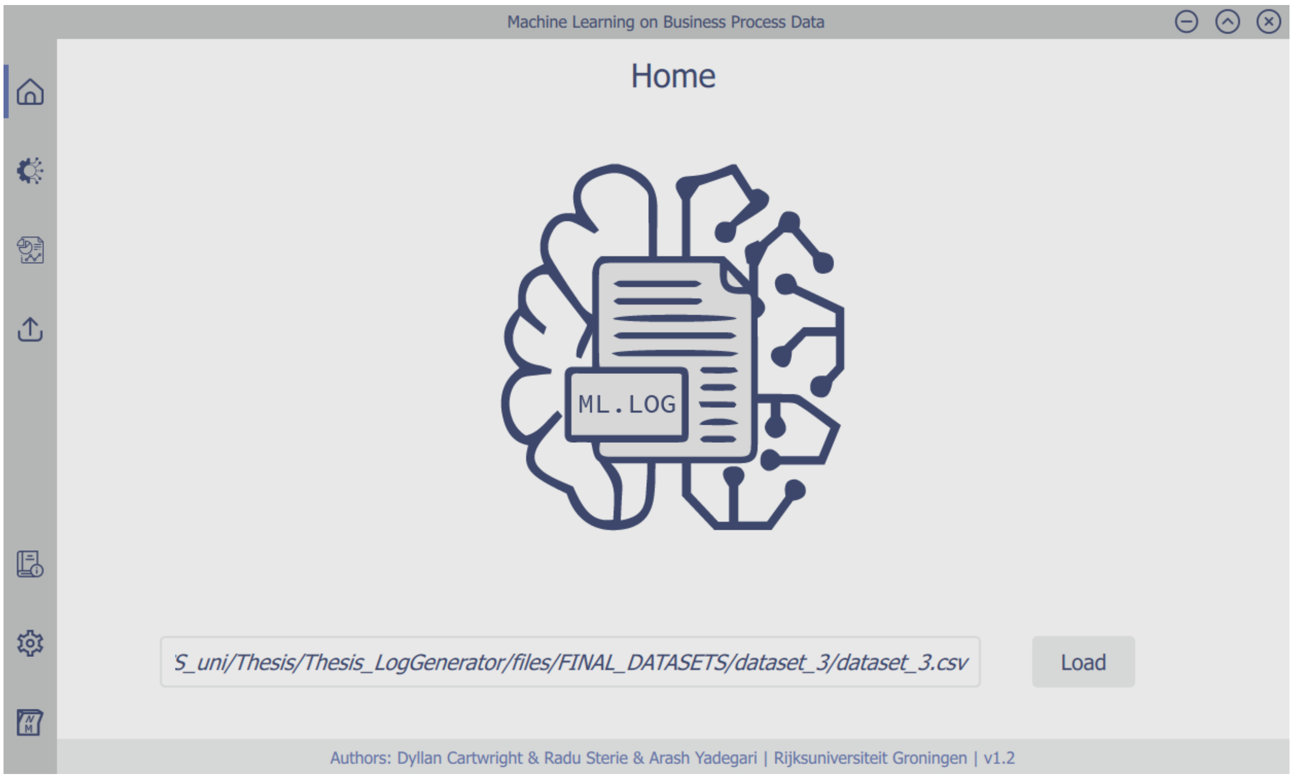
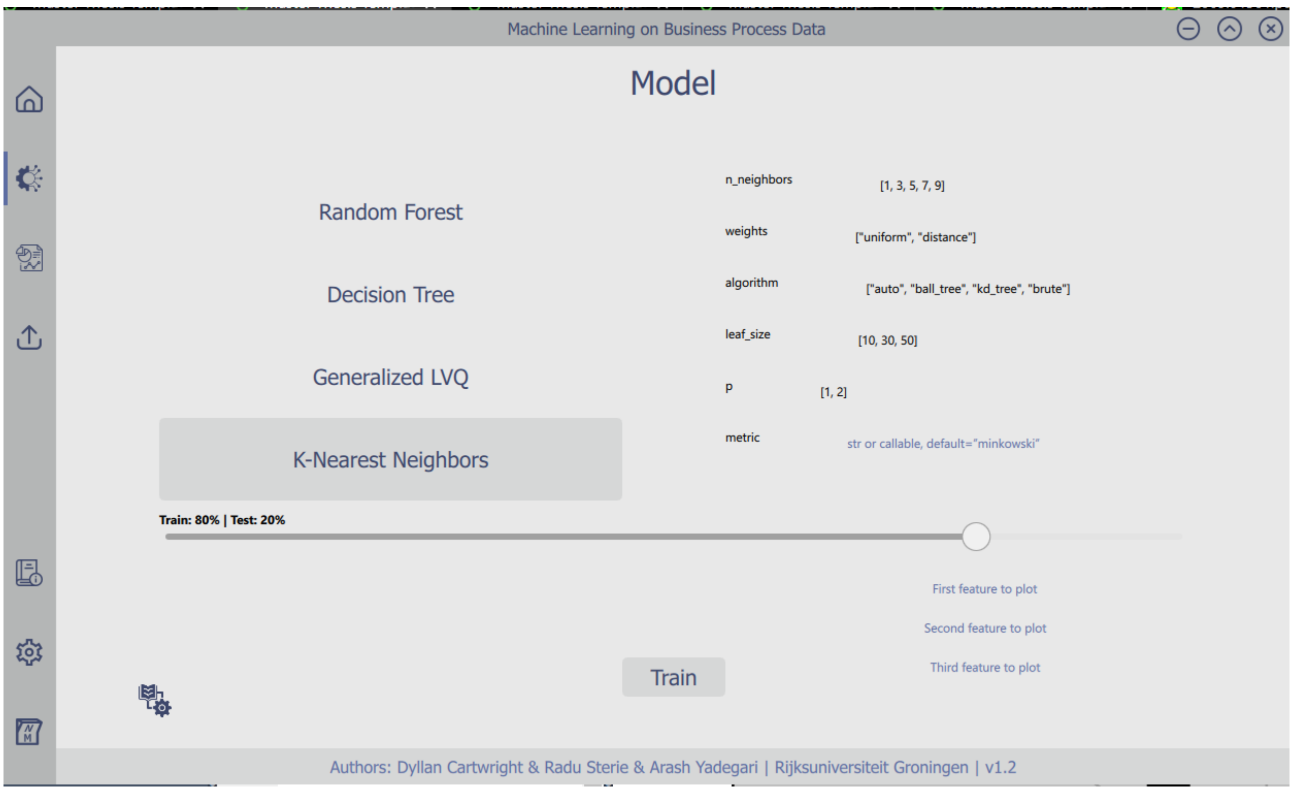

Machine Learning on Business Process Data
Home
ML.LOG
S_uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/dataset_3.csv
Load
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 20: Home Page


Machine Learning on Business Process Data

Model

Random Forest

Decision Tree

Generalized LVQ

K-Nearest Neighbors

n_neighbors	[1, 3, 5, 7, 9]
weights	["uniform", "distance"]
algorithm	["auto", "ball_tree", "kd_tree", "brute"]
leaf_size	[10, 30, 50]
p	[1, 2]
metric	str or callable, default="minkowski"

Train: 80% | Test: 20%

First feature to plot

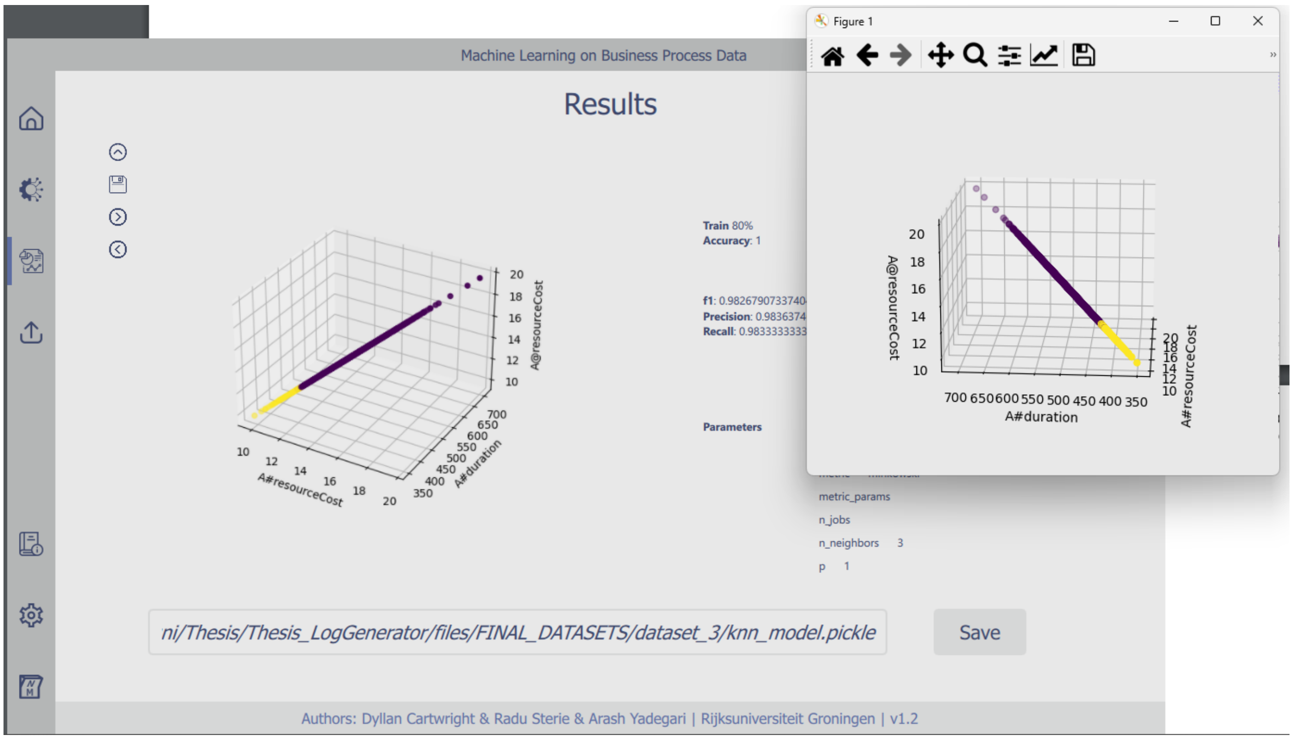
Second feature to plot

Third feature to plot

Train

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 21: Select Model Page


Machine Learning on Business Process Data

Results

Train 80%
Accuracy: 1

f1: 0.9826790733740
Precision: 0.9836374
Recall: 0.9833333333

Parameters

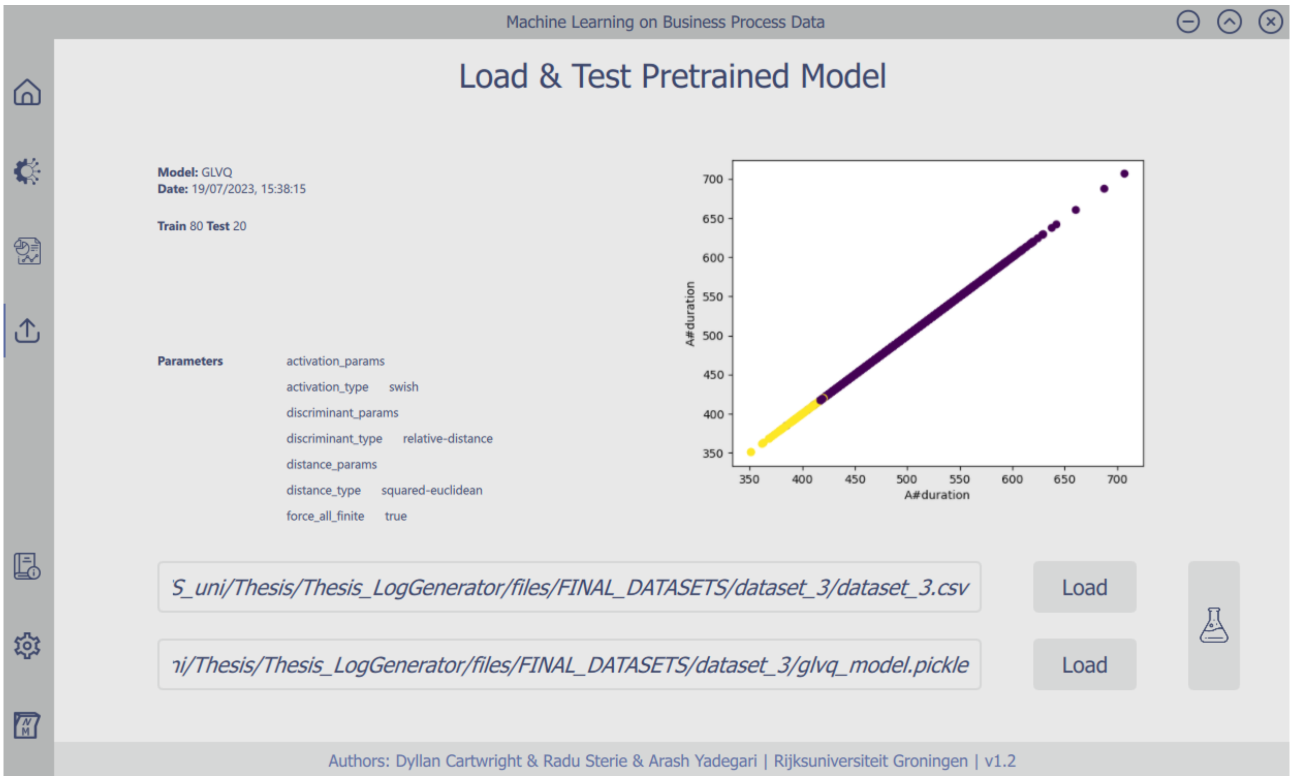
metric_params	
n_jobs	
n_neighbors	3
p	1

ni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/knn_model.pickle

Save

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 22: Results Page


Machine Learning on Business Process Data
Load & Test Pretrained Model
A#duration
A#duration
Load
Load
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Model: GLVQ
Date: 19/07/2023, 15:38:15

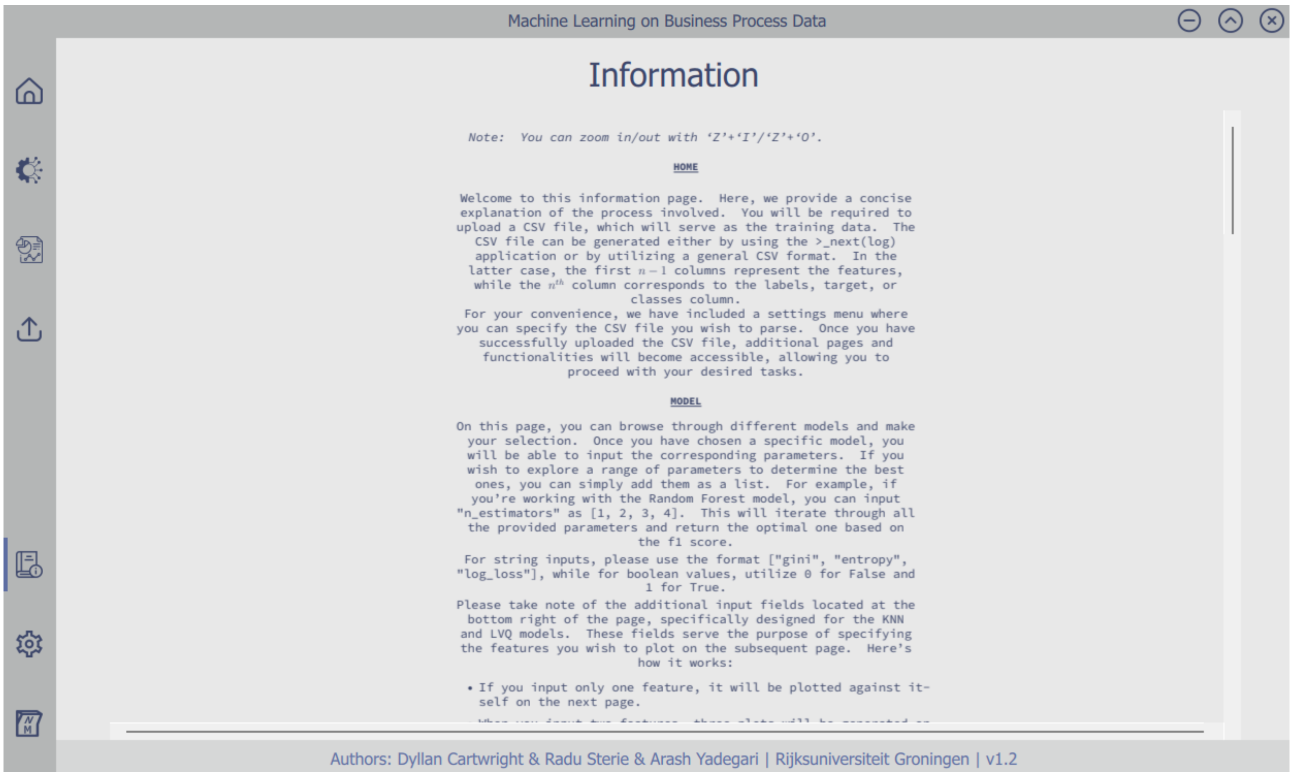
Train 80 **Test** 20

Parameters		
	activation_params	
	activation_type	swish
	discriminant_params	
	discriminant_type	relative-distance
	distance_params	
	distance_type	squared-euclidean
	force_all_finite	true

S_uni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/dataset_3.csv

ni/Thesis/Thesis_LogGenerator/files/FINAL_DATASETS/dataset_3/glvq_model.pickle

Figure 23: Load & Test Model Page


Machine Learning on Business Process Data
Information
Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Note: You can zoom in/out with ‘Z’+‘I’/‘Z’+‘O’.

HOME

Welcome to this information page. Here, we provide a concise explanation of the process involved. You will be required to upload a CSV file, which will serve as the training data. The CSV file can be generated either by using the >_next(log) application or by utilizing a general CSV format. In the latter case, the first $n-1$ columns represent the features, while the n^{th} column corresponds to the labels, target, or classes column.

For your convenience, we have included a settings menu where you can specify the CSV file you wish to parse. Once you have successfully uploaded the CSV file, additional pages and functionalities will become accessible, allowing you to proceed with your desired tasks.

MODEL

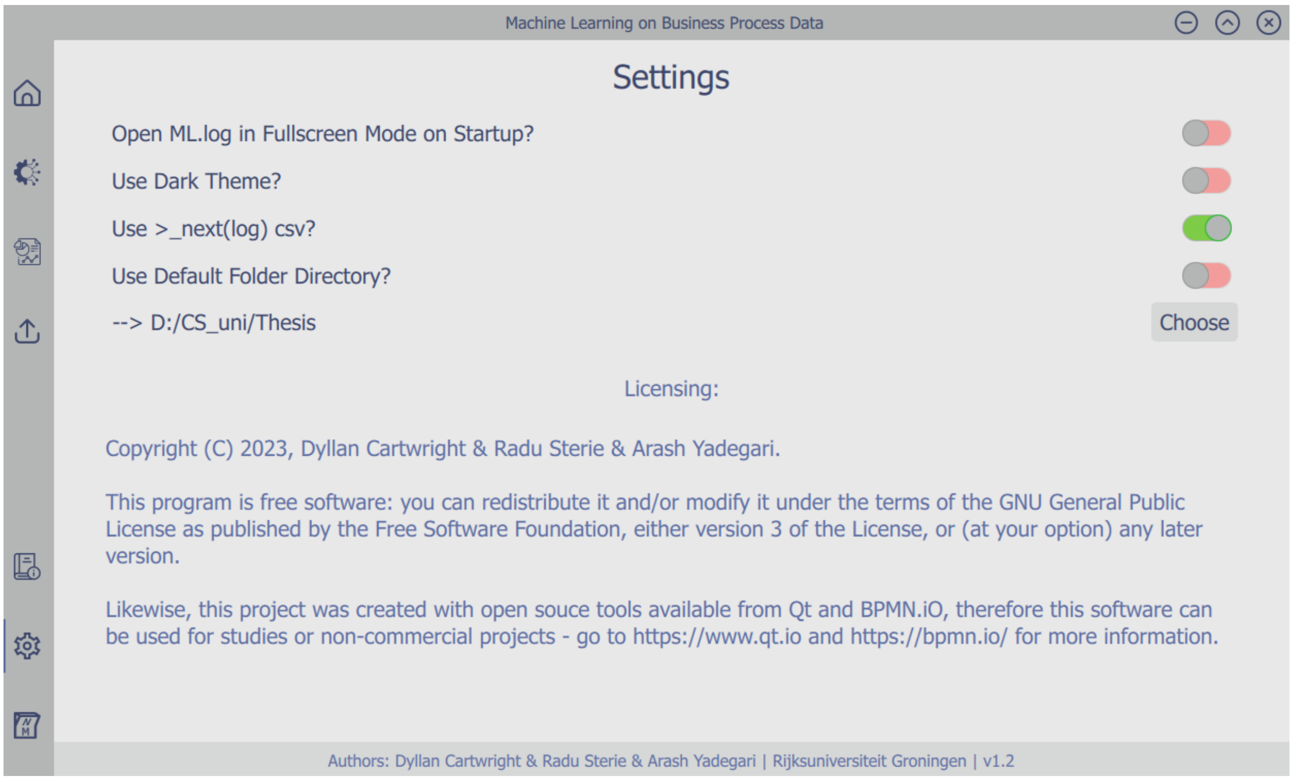
On this page, you can browse through different models and make your selection. Once you have chosen a specific model, you will be able to input the corresponding parameters. If you wish to explore a range of parameters to determine the best ones, you can simply add them as a list. For example, if you’re working with the Random Forest model, you can input "n_estimators" as [1, 2, 3, 4]. This will iterate through all the provided parameters and return the optimal one based on the f1 score.

For string inputs, please use the format ["gini", "entropy", "log_loss"], while for boolean values, utilize 0 for False and 1 for True.

Please take note of the additional input fields located at the bottom right of the page, specifically designed for the KNN and LVQ models. These fields serve the purpose of specifying the features you wish to plot on the subsequent page. Here’s how it works:

- If you input only one feature, it will be plotted against itself on the next page.

Figure 24: Info Page


Machine Learning on Business Process Data

Settings

Open ML.log in Fullscreen Mode on Startup? — off

Use Dark Theme? — off

Use >_next(log) csv? — on

Use Default Folder Directory? — off

--> D:/CS_uni/Thesis — Choose

Licensing:

Copyright (C) 2023, Dyllan Cartwright & Radu Sterie & Arash Yadegari.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Likewise, this project was created with open souce tools available from Qt and BPMN.iO, therefore this software can be used for studies or non-commercial projects - go to https://www.qt.io and https://bpmn.io/ for more information.

Authors: Dyllan Cartwright & Radu Sterie & Arash Yadegari | Rijksuniversiteit Groningen | v1.2

Figure 25: Settings Page