Universiteit Leiden

Master Computer Science

Enhancing the Reliability of Model Evaluations in Predictive Process Monitoring

Name:	Hidde van Rooijen
Student ID:	3377954
Date:	26-05-2025
Specialisation:	Data Science
1st supervisor:	dr. Peter van der Putten
2nd supervisor:	Prof. dr. Hans Mulder

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Predictive Process Monitoring (PPM) is an upcoming research field aimed at developing machine learning models to predict various aspects of business processes. For training these models, researchers rely on process event logs commonly used for process mining. There are a variety of publicly available event logs that researchers can choose from to benchmark their models.

However, recent publications have shown that commonly used evaluation methods and datasets produce unreliable estimates of real-world performance, due to data leakage and other forms of bias present in event logs. Furthermore, the lack of standardised evaluation methods makes it difficult to compare models between different studies. Researchers use varying datasets, preprocessing techniques, and evaluation metrics, preventing meaningful comparisons between approaches.

For this thesis, a benchmark framework was developed to address these challenges. Following a design science approach, the following features were implemented: a debiasing strategy that removes selection bias from datasets, automated baseline predictions for proper comparison, visualisation tools that reveal model behaviour beyond simple metrics, and the Probability Ratio Score (PRS) metric to measure model performance in ambiguous situations. The framework was built to be easily extensible; researchers can add new datasets and metrics without changing the core structure.

A literature review identified 111 articles, of which 27 had available source code. Three algorithms were ultimately implemented and tested: ProcessTransformer, PGTNet, and PyDREAM. The results show significant performance drops on debiased datasets compared to reported scores. For remaining time prediction, no model outperformed a simple median baseline. For next activity prediction, while ProcessTransformer achieved the best scores, a simple distance-based baseline performed surprisingly well. The visualisations revealed insights impossible to capture with metrics alone, showing that some models only excel on cases with activity sequences already seen during training.

This work contributes a practical tool for more reliable PPM evaluation. The framework is available on GitHub to encourage community adoption and extension.

Contents

Chapter 1

Introduction

Process mining is an emerging field that lives on the intersection of data science and process science. The technology leverages (real-time) process data to construct process models, which allow for a detailed evaluation of business processes. Due to the increasing popularity of process mining, a new field has emerged which leverages the datasets used for process mining to make predictions for intelligent decisions in business processes using machine learning techniques.

Several models have been developed that aim to predict various aspects of business processes such as the next activity, the remaining time, or the result of a running process [Rama-Maneiro et al., 2021, Teinemaa et al., 2019]. The models described in the literature are usually trained and evaluated on publicly available process datasets. These data sets are divided into a training set and a test set, where the test set is used to evaluate the model. Splitting the data set in this way is common practice in many machine learning fields and enables researchers to estimate a model's real-world performance.

However, recent publications have shown that evaluating predictive process monitoring models in this way will not produce reliable estimations of a model's real-world performance [Weytjens and De Weerdt, 2021, Abb et al., 2023]. This is due to large amounts of data leakage and other types of test set bias. This is mainly due to the fact that a large portion of process executions tend to be identical, resulting in a test set with samples already observed in the training data. Furthermore, even if performance estimates were reliable, researchers tend to use varying datasets and evaluation metrics, making it difficult to compare models. There is no standardised method for reliable model evaluations within the field.

The motivation for this thesis stems from the need to improve the reliability and comparability of PPM model evaluations, which is crucial to advance the field and ensure that models can be effectively applied in real world settings. This will be done by developing a benchmark framework with improved and standardised evaluation techniques.

1.1 Research questions

To address the challenges outlined above, this thesis investigates the following research questions.

RQ 1: *Which methods can be further developed which can effectively measure the generalization ability of predictive process monitoring models across diverse generalization scenarios?*

RQ 2: *What models can be found in PPM literature for each of the prediction tasks found in literature?*

RQ 3: *What are the software requirements for a benchmark suite which allows for easy integration of the methods designed in RQ 1, and allows for the addition of new methods in the future?*

RQ 4: *What is the performance of the identified state-of-the-art models (RQ2) on the constructed benchmark, and how does this compare to the performance described by the authors?*

The remainder of this thesis is structured as follows. Chapter 2 provides the necessary technical background and reviews related work, detailing current model training approaches and the specific evaluation challenges arising from event log data. Chapter 3 will describe the methodology used to answer each of the research questions, including the software requirements and the benchmark framework. Chapter 4 contains the details of the implementation of the implemented framework. With the methodology and benchmark framework details covered, chapter 5 will continue with the validation of the framework. The chapter describes the experimental setup, which includes motivation on model selection, and benchmark results on two prediction problems for the selected models. Chapter 6 will discuss the results in more detail, including the limitations of this study, and provide future research directions. Chapter 7 contains the conclusion of the results and the performed work.

Chapter 2

Background & Related Work

The following chapter will provide the reader with the necessary background knowledge to understand the relevance of the research performed. It also provides an overview of previous work done in the field that served as an inspiration for this thesis.

2.1 Process mining

Process mining is a technology that lives at the intersection of data science and process science. Using event data generated during the execution of a process, process mining algorithms construct BPMN models of the true execution of the process. These models can then be used to detect issues in a (business) process. Common use cases are: detecting compliance violations, identifying bottlenecks, root cause analysis, or performing predictive process monitoring. [van der Aalst, 2022].

All process mining algorithms require event data to build process models. This event data is usually a tabular data set that contains a large number of cases (process executions) including the activities performed for each case and its timestamp [van der Aalst, 2022].

Table 2.1 shows an example of such event data for a simplified online purchase process. This process includes the activities 'placing an order', 'payment processing', and 'order shipment'. The event log contains a timestamp at which the activity was executed, the resource that executed the activity, and to which case these attributes belong.

Table 2.1: Sample event log for online purchase process.

Case ID	Timestamp	Activity	Resource
1	2024-04-18 08:00:00	Placing an Order	Website
1	2024-04-18 08:05:00	Payment Processing	Payment Sys.
1	2024-04-18 08:15:00	Order Shipment	Warehouse
2	2024-04-18 09:00:00	Placing an Order	Website
2	2024-04-18 09:03:00	Payment Processing	Payment Sys.
2	2024-04-18 09:20:00	Order Shipment	Warehouse
3	2024-04-18 09:30:00	Placing an Order	Website
3	2024-04-18 09:35:00	Payment Processing	Payment Sys.
3	2024-04-18 09:45:00	Order Shipment	Warehouse

2.2 Predictive process monitoring

The event logs created for process mining also enable practitioners to perform predictive analytics on their process data. This field of predictive process monitoring (PPM) has gained a lot of popularity over the past few years. Using machine learning techniques, researchers aim to predict a variety of attributes or outcomes for ongoing cases. Common prediction types include next activity prediction, remaining time prediction, or categorical outcome prediction [Rama-Maneiro et al., 2021].

2.2.1 Model training

Within predictive process monitoring, a variety of prediction tasks can be identified. Rama-Maneiro et al. [Rama-Maneiro et al., 2021] performed a thorough analysis of current deep learning methods in the field of predictive process monitoring and identified six main prediction tasks. These prediction tasks are:

- **Next Activity Prediction:** Predicts the next activity in a business process based on historical event log data.
- **Next Attribute Prediction:** Predicts the next attribute of an event.
- **Next Timestamp Prediction:** Predicts the time until the next activity occurs.
- **Remaining Time Prediction:** Predicts the time remaining until the completion of a process instance.
- **Activity Suffix Prediction:** Predicts the sequence of future activities that will occur until the case is completed.
- **Attribute Suffix Prediction:** Predicts the sequence of future attributes for the rest of the case.
- **Outcome Prediction:** Forecasts the outcome of a running case or a process instance.

Various methods exist to train predictive process monitoring models. Some researchers leverage deep learning models which support the direct input of process sequences with minimal data preprocessing. Others leverage a variety of sequence encoding techniques to transform an event log into a machine learning ready format. Methods for encoding event sequences for deep learning include [Rama-Maneiro et al., 2021]:

- **Continuous (prefix padded):** Inspired by language models, treats the event log as a continuous sequence. Can be implemented either using a sliding window or using the entire prefix. Padding might be used to create sequences of equal length.
- **N-gram:** Breaks down the prefix into all possible sub-sequences of size k. Meaning an n-gram encoded event log is represented as a set of all sub-sequences of length k contained within it.
- **Single event:** A simple encoding method that only considers a single event and its attributes as input.

- **Timed state:** An encoding method that captures not only the sequence of events, but also the timing between them. It uses a decay function to represent the state of the process, incorporating how long ago events occurred and the frequency of transitions.

Apart from the single-event encoding, these methods result in a sequence which will serve as input to the model. Classical machine learning algorithms do not support the input of such sequences and thus require different encoding techniques. These methods aim to "flatten" the event log into a tabular dataset. These methods include [Teinemaa et al., 2019]:

- **Last state:** Only use the last state of the process and its attributes as input. Can be combined with aggregation to create more meaningful features.
- **Aggregation:** Aggregate attributes from the trace into a single row. For example, averaging numerical columns or taking the most common string.
- **Index-based:** Transforming the event log to include a column for each event attribute and its index.

All these different sequence-encoding schemes show the differences in philosophy throughout the literature. Some encoding techniques aim to keep as much information from the original sequence in the training data as possible (index-based , continuous, timed state), assuming long-range dependencies within the data. Others disregard the idea of long-range dependencies and leverage techniques based on single events.

2.3 Model evaluation

In the field of predictive process monitoring, model evaluation is usually performed on publicly available real-life event logs. These are often event logs from the Business Process Intelligence Challenge (BPI), which was a process mining challenge organised during the yearly International Conference on Process Mining (ICPM). Although data sets are usually the same, evaluation methods are usually different between research papers. This section will proceed with some examples of how model evaluation is done for a small, random selection, of articles published in the field.

The benchmark performed in [Rama-Maneiro et al., 2021] performs 5-fold cross-validation, where the final performance is given as the performance in the final fold. The splitting is done in an 80/20 ratio. Events are ordered using the timestamps from the trace. Metrics used for evaluation include: accuracy (for next activity), Damerau-Levenshtein distance (for activity suffix) or Mean Absolute Error (for next- and remaining time). Although this paper describes the prediction tasks outcome, attribute and attribute suffix, they do not consider these prediction problems in their benchmark. The paper does not describe any event log-specific transformations done. No details are given for the hyperparameter search space in the paper. The code is available on GitHub, but a quick analysis shows no systematic approach for defining the hyper-parameter search space.

Teinemaa et al. [Teinemaa et al., 2019] performed a benchmark on outcome-orientated predictive process monitoring approaches. The authors compare various algorithms and combinations of sequence encoding techniques. They performed a three-fold cross-validation and used the AUC metric to determine the best model. The paper also provides

metrics for computation time and earliness of the prediction. Since the authors focused on outcome-oriented predictions, the event logs were enhanced with custom labels to simulate the process outcomes.

In 2021 Kratsch et al. performed another comparison of outcome-oriented approaches. They used only a partial subset of the logs in [Kratsch et al., 2021] and created their own set of outcome labels. The authors performed a 10-fold cross-validation on each of the compared methods. Performance is given using the following metrics: accuracy, precision, recall, F-beta, and AUC score.

2.4 Challenges in model evaluation

Most of the studies described in section 2.3 evaluate models by performing a train-test split and/or performing k-fold cross-validation. Splitting is commonly done by sorting events according to their occurrence time in the event log, and taking the first n rows for training and the remaining part for evaluation. This is standard practice in machine learning research, as it makes sure the model is evaluated on unseen data. However, recent publications have shown that evaluations done in this matter might not always produce reliable estimates of a model's real-world performance [Abb et al., 2023].

2.4.1 Generalization

Abb et al. [Abb et al., 2023] have published a discussion on example leakage and generalization in next activity prediction. The authors argue that current evaluation methods produce unreliable estimates of the real-world performance of a model. The authors identify a problem where most PPM models learn to predict the next activity primarily based on the already completed activities. This questions whether PPM models actually learn anything about the underlying business process, or completely rely on the activity prefix, which contains high levels of example leakage.

This concept is demonstrated by comparing a state-of-the-art deep learning model with a naive baseline, which only predicts the most common next activity based on the last activity. This comparison is done on five popular benchmark event logs. The results show only a few percentage points difference in performance for the worst-case scenario and the same performance in the best-case scenario. The same performance is achieved on an event log where the test set contains no new activity traces compared to the training set. The question is whether this small difference in performance is caused by the fact that this is simply a trivial prediction task, and thus both models being close to the maximum achievable score, or whether there exists a more complex relation which is not captured due to overfitting on the activity prefix.

The authors also discuss an 'accuracy limit' problem found in nearly all event logs. Consider two identical activity prefixes that precede some decision point in a process. For a naive model, there is a clear maximum achievable accuracy for this type of path as it will always predict the most common 'decision'. The authors show that state-of-the-art models are also bounded by this same accuracy limit. Intuitively, one might argue that more complex models should be able to learn at least some of the factors influencing the decision based on other process attributes, thus being able to exceed this maximum as its only determined by the activity prefix. This, combined with the observation that most event logs contain high levels of example leakage, calls into question the level of generalisation these state-of-the-art models are capable of.

The authors continue by describing three generalisation tasks which could be measured to determine the generalisation ability of next activity prediction models. These are: unseen control flow, unseen attribute value combinations, and unseen attribute values. An unseen control-flow can occur when the training data does not contain an example of a specific sub-sequence of activities. This can happen if a subset of process activities is allowed to be executed in any order, and this particular sequence was not present in the training data. I.e. activities C1, C2, and C3 can occur in any order, and the sequence $\langle C1, C3, C2\rangle$ was not part of the training data. A model capable of generalisation should have learnt that the subsequence has completed and that the next activity is D.

Unseen attribute value combinations occur when the training data does not contain an example of a specific activity attribute pair. An example of this is when the training data only contains traces in the form of $\langle (A, R1), (B, R2), (C, R3)\rangle$, and a new trace $\langle (A, R1), (B, R1)\rangle$ is given. A model capable of generalisation should predict C, as C always follows B. The model should not take into account the resource associated with B to make the prediction in this case.

Sometimes, training data might not contain all unique attribute values. This can happen, for example, if new resources are employed or product prices change. Predictive process monitoring models should be capable of handling such changes and generate informed predictions. I.e. if some numeric attribute influences the path of the process and a new numeric value is observed, the model should predict the path of the closest numeric value in the training data.

The generalisation challenges described above are merely a starting point and do not cover all prediction tasks. More work is needed to come up with more ways to determine PPM model's generalisation capabilities and implement methods to measure these.

2.4.2 Selection bias

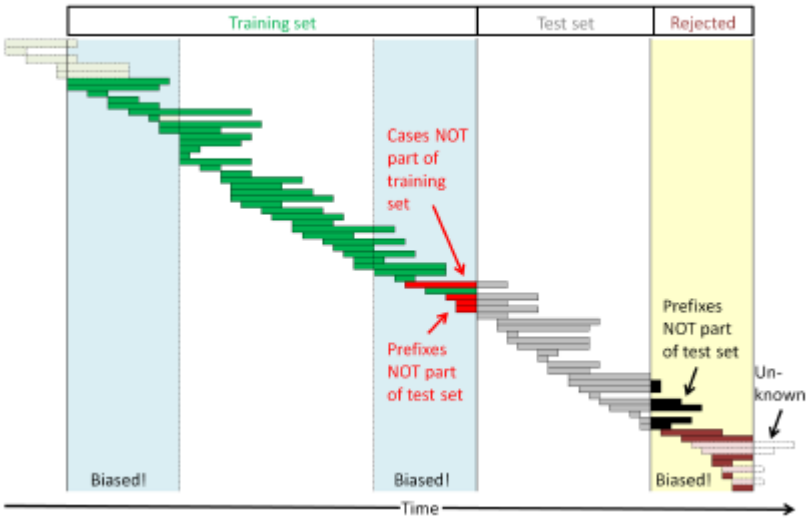
Weytjens and de Weerdt [Weytjens and De Weerdt, 2021] have published paper describing two forms of selection bias that occur when splitting data using the methods described earlier, specifically for predicting the remaining time. Firstly, in most cases, to be able to determine the target variable for prediction, one must filter the event log for completed cases. This produces a form of bias where uncompleted long-running cases are removed from the end of the dataset, resulting in a test set which contains a larger proportion of short cases compared to the training data.

The second form of bias comes from the inter-case properties at the end of this filtered event log. Since uncompleted cases are removed from the data, the number of running cases at the end of the data set no longer reflects the underlying business process. If this last part of the dataset is used for testing, any learnt inter-case dynamics are not usable for the test set.

The authors propose a method to debias the end of the event log, by removing all events that occur in a time window equal to the longest duration case from the end. A similar process is followed to split the training data from the test data. Figure 2.1 shows a visual representation of which parts of the dataset are removed. In Section 4.1 of this thesis you will find more details on the implementation of this debiasing strategy.

Some work has been done by the authors to create benchmark data sets that remove this type of bias. However, their public Github repository shows only a limited amount of datasets, which could be further extended.

Figure 2.1: Visual representation of testset debiasing strategy introduced by Weytjens and de Weerdt [Weytjens and De Weerdt, 2021]



Chapter 3

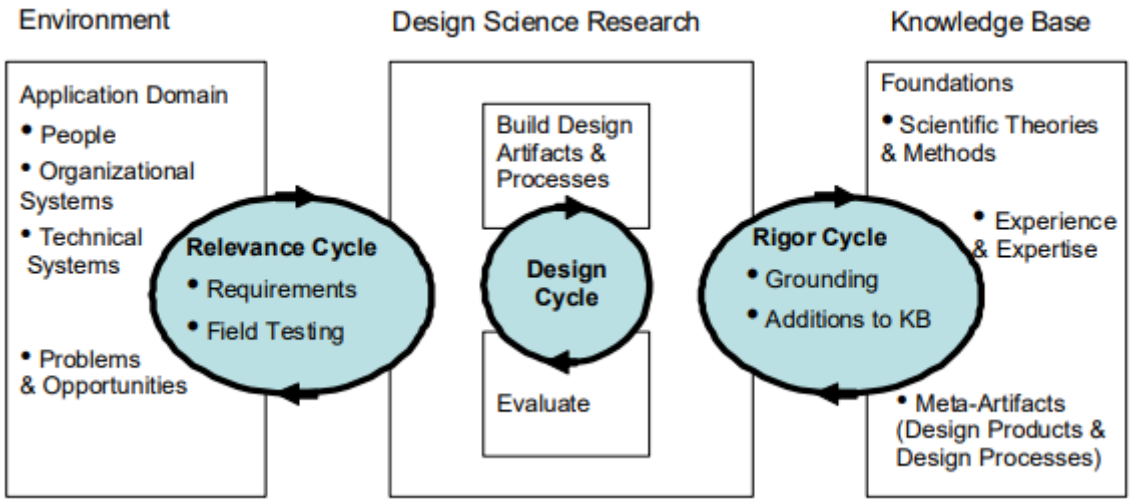
Methodology

This chapter provides a detailed description of the methodology used for this research. These are given as the six design science objectives.

3.1 Design science

This research will follow a design science approach. This is a research paradigm that focusses on the development and evaluation of artefacts that aim to improve an environment. In this case, the development of a reusable benchmark framework for the evaluation of PPM models. Hevner describes three main cycles in design science research [Hevner, 2007], these are outlined in Figure 3.1. These cycles are the relevance cycle, the rigour cycle, and the design cycle.

Figure 3.1: The three design science cycles



Peffers et al. [Peffers et al., 2020] identified six main objectives which must be included in the design science research of information systems. These are:

1. Problem identification and motivation
2. Objectives of a solution
3. Design and development

4. Demonstration
5. Evaluation
6. Communication

These objectives can be translated into the design science cycles described by [Hevner, 2007]. Objectives 1 and 2 are part of the relevance cycle. The design cycle encompasses objectives 3-5, while objective 6 is grounded in the rigour cycle. To ensure that this research produces useful contributions to the field of predictive process monitoring, I will further elaborate the methodology of this research based on the principles described above.

3.2 Problem identification

Problem identification was done before starting the thesis project. This was done by writing a comprehensive research plan that included relevant related work and possible contributions to the field. This document contains the most important details for this problem identification.

3.3 Solution objectives

The long-term objective of this research is to reach a consensus on model evaluation methods in the field of PPM. It is naive to think that a single master's thesis can reach this objective. Moreover, as more prediction tasks are developed and more public event logs become available, the requirements for such an evaluation approach might shift. Therefore, it is important to design an approach that can withstand changing requirements (RQ3). This makes another objective of this research to design a design science method itself.

3.3.1 The model evaluation framework

In practice, a carefully designed, open-source, extendable framework can serve as a design science artefact. As the objective is to build a software package for PPM model evaluation, the solution objectives for this part will be given in terms of requirements. This is a common practice in software development. These are very general requirements and require further refinement. Requirement refinement will be an ongoing process during this research as more knowledge becomes available. The following is a list of initial (rough) requirements.

1. The solution should allow the user to retrieve a test- and training dataset using an API.
2. The solution should have a consistent API regardless of the data set or evaluation metric.
3. Adding new evaluation datasets should be straightforward, maintaining a consistent API.
4. Adding new evaluation metrics should be straightforward and maintain a consistent API.

5. A knowledge base should be part of the final solution containing: model evaluations with paper references, documentation on how to evaluate models, and documentation on how to extend the framework.
6. Must be built in Python
7. Framework must work with models trained using: Scikit-learn, Tensorflow, or pyTorch.
8. The data sets must be downloadable either using the library or included directly with the source code.
9. After evaluating the model, a report should be returned to the user containing the following evaluation metrics:
    - For general classification tasks: AUC, precision, recall, F-1 and accuracy
    - For sequence classification tasks: Damerau-Levenshtein distance
    - For regression tasks: MSE, RMSE, and MAE

Model evaluation techniques
To develop new PPM model evaluation techniques, RQ1 must be answered. This requires the development of evaluation methods that capture the generalisation ability of models described in the article by Abb et al [Abb et al., 2023]. This will be done by constructing various benchmark datasets that can be used to measure a model's generalisation ability. The following event log characteristics will be taken into account for evaluating models and measuring generalisation:

- **Unseen activity sequences:** measure the performance of the model when including unseen activity sequences at various levels of complexity.
- **Label ambiguity:** Performance measurements should account for potentially high levels of label ambiguity in event logs.
- **Attribute drift:** measure the performance of the model when various types of attribute drift are introduced, such as new resources or new activities.

3.4 Design and development

The design and development cycle will consist of designing a system architecture and refining the requirements. Part of the input for this cycle will be the answers to RQ1, as this is needed to refine requirements related to specific generalisation metrics.

At least three iterations of the design cycle can be identified when considering the requirements of Section 3.3. Firstly, a 'skeleton framework' must be developed. This skeleton framework is based on requirements 1-4. After that, the skeleton framework can be used to implement the evaluation methods identified by answering RQ1. The third cycle will consist of improvements as the framework is rigorously tested during the answering of RQ4.

Completing these cycles will both ensure a robust framework capable of handling shifting requirements as the field of PPM matures, and provide an initial contribution to solving the problems described in sections 2 and 3.

3.5 Demonstration and evaluation

Demonstration & evaluation of the framework will be carried out by answering RQ4 and evaluating the design phases described in Section 3.4. The framework will be considered as a significant contribution if the skeleton framework is sufficient to extend the framework with new evaluation metrics developed after answering RQ1. Evaluation methods will be considered successful if they successfully capture differences in state-of-the-art models and the naive baseline described in [Abb et al., 2023].

3.5.1 Demonstration of evaluation framework

To answer RQ4, which aims to demonstrate the developed framework, RQ2 needs to be answered. This requires the identification of relevant models in the literature. This will be done by performing a backwards snowball literature review. Snowball literature research involves selecting an initial, relevant paper and then systematically exploring its references. This process is repeated recursively, following citations within each referenced paper, until a predetermined stopping criterion is met, such as reaching a saturation point where no new relevant information is found or when the sources become too outdated or irrelevant. The starting paper for this literature research will be the work of Weinzierl et al. [Weinzierl et al., 2024] as it is the most recent published literature review related to the field.

Two general rules will be followed for including sources for further exploration: the topic of the source must be a literature review on PPM or a novel PPM method, and the source must be published after January 2019. The first rule will prevent a complete 'explosion' of sources while still allowing more exploration using literature reviews. The second rule will save some work in analysing papers of outdated methods. It is likely that any method published before 2019 is somewhat outdated. It is also not the main goal of this review to provide a complete overview of all published methods, as the only goal is to validate the developed benchmark.

3.6 Communication

Communication is part of the knowledge base component of the design cycle in Figure 3.1. Section 3.3 describes the requirements for this knowledge base as part of the general requirements of the framework. Communication will be done in two-fold: a complete master's thesis describing the proposed solution and a comprehensive documentation of the solution.

The thesis will be mainly geared towards the description and evaluation of all objectives set in Section 3.3. As one of the objectives is to create a framework that allows for further extension, there must also be a standalone documentation/knowledge base for the framework. This knowledge base should contain information on how to extend the framework, use it to evaluate models, and contain the evaluations of state-of-the-art models. All of this can be found on the GitHub page of this thesis[1].

[1] https://github.com/hiddevr/ppm_benchmark

Chapter 4

Implementation

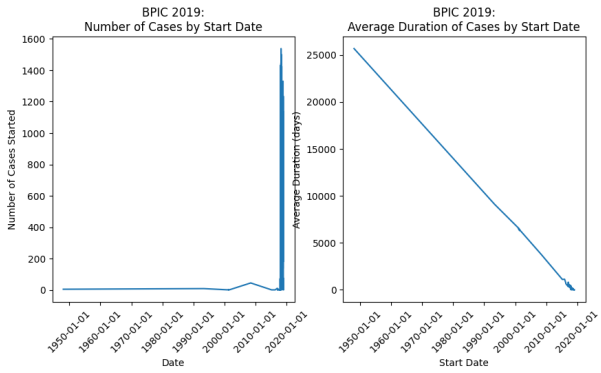
Chapter 4 describes the technical details of the implemented benchmark framework. It includes a detailed UML diagram, motivation behind the dataset selection, and the strategy used to measure generalization in PPM models. For more details on the benchmark framework, see the GitHub project[1].

4.1 Benchmark data selection

Popular datasets for evaluating PPM models include the Business Process Intelligence Challenge (BPIC) event logs [van Dongen, 2020b], hospital billing event log [Mannhardt, 2017] and sepsis cases event log [Mannhardt, 2016]. These are all datasets from real business processes. As mentioned earlier, the data in the benchmark will be debiased using the methods proposed in [Weytjens and De Weerdt, 2021].

The debiasing strategy works by first identifying any chronological outliers in a dataset. These are cases that start significantly earlier/later compared to all other cases in the event log. This is done by performing a visual inspection of the number of cases by start date. Figure 4.1 shows an example plot used for visual inspection. We can clearly see that any cases that start before the main peak (around 2018) can be considered outliers. The second step in the debiasing process is debiasing the end of the dataset. This is done by removing all cases from the end that are longer than the case of longest duration (after removing the top 5%). This ensures that only finished cases remain in the dataset, removing any duration bias from unfinished shorter cases at the end of the data set.

Figure 4.1: Example of dataset containing chronological outliers.



[1] https://github.com/hiddevr/ppm_benchmark

After that, a strict temporal split is applied on the dataset, meaning only cases completed before the separation time are included in the train set. For this benchmark framework, a test set size of 20% was chosen. Cases that started before the separation time but ended after the separation time are added to the test set. This leaves a slightly biased part at the end of the train set and the start of the test set, for which no solution has been found yet.

Table 4.1 shows all datasets selected for the benchmark framework with their properties after applying the debiasing strategy described above. It appears that not all of the datasets are suitable for PPM model benchmarking, as insufficient samples remain after debiasing. This result underlines the importance of the debiasing strategy and suggests that existing publications in the field might rely too heavily on biased datasets.

The datasets BPIC 2011, all BPIC 2013 datasets, Mobis and Production all lose too many cases after debiasing to be representative of the original business process. The BPIC 2016 data set is a special case where the data set will be completely removed when applying the debiasing strategy. This is likely caused by the fixed time span over which the event log was created, resulting in nearly all cases being unfinished.

Table 4.1: Dataset properties of the identified publicly available event logs after applying the debiasing strategy.

Dataset	Top 5% duration (days)	# Cases original	# Cases train	# Cases test	% Cases retained	Date start	Date end	Reference
BPIC 2011	992	1143	31	210	21,09%	-	-	[van Dongen, 2011]
BPIC 2012	31,34	12087	7799	2719	87,03%	-	-	[van Dongen, 2012]
BPIC 2013: Open problems	324,95	716	280	88	51,39%	2011-10	-	[Steeman, 2014]
BPIC 2013: Closed problems	660,71	1456	221	70	19,98%	2010-01	-	[Steeman, 2014]
BPIC 2013: Incidents	35,7	7228	563	3155	51,41%	2012-04	-	[Steeman, 2014]
BPIC 2014	19,99	46616	31582	8675	86,39%	2013-10	-	[van Dongen, 2014]
BPIC 2015: Municipality 1	313,43	1199	711	242	79,48%	2010-10	-	[van Dongen, 2015]
BPIC 2015: Municipality 2	470,37	832	407	177	70,19%	2010-10	-	[van Dongen, 2015]
BPIC 2015: Municipality 3	197	1409	914	268	83,90%	2010-10	-	[van Dongen, 2015]
BPIC 2015: Municipality 4	274,65	1053	579	243	78,14%	2010-10	-	[van Dongen, 2015]
BPIC 2015: Municipality 5	260,48	1156	721	212	80,66%	2010-10	-	[van Dongen, 2015]
BPIC 2016	235,06	27412	-	-	-	-	-	[Dees and van Dongen, 2016]
BPIC 2017	42,52	31509	21314	7854	92,62%	-	-	[van Dongen, 2017]
BPIC 2019	142,3	251734	85277	86055	68,02%	2018-01	2019-02	[van Dongen, 2019]
BPIC 2020: Domestic declarations	29,2	10500	7692	2186	94,17%	-	-	[van Dongen, 2020a]
BPIC 2020: International declarations	244,35	6449	2525	2288	74,61%	2017-01	-	[van Dongen, 2020a]
BPIC 2020: Payments	31,27	6886	5035	1486	94,68%	2017-01	-	[van Dongen, 2020a]
BPIC 2020: Permits	209,32	7065	3952	2555	92,17%	2017-01	-	[van Dongen, 2020a]
BPIC 2020: Travel Costs	112,17	2099	1263	545	86,14%	-	-	[van Dongen, 2020a]
Helpdesk	57,99	4580	3221	1084	93,93%	-	-	[Polato, 2017]
Mobis	168,17	6555	665	2016	40,89%	2017-02	-	[Scheid et al., 2018]
Production	72	225	-	-	-	14-01-2012	-	[Levy, 2014]
Traffic	968	150370	89687	31476	80,56%	2012-12	-	[de Leoni and Mannhardt, 2015]

4.2 Measuring generalization

Section 2.4.1 has introduced some of the work done by Abb et al. [Abb et al., 2023] on defining what generalization means in the context of next activity prediction in PPM. This section will expand on this work by proposing methods to measure generalization for each prediction task described in section 2.2.1. The goal is to find a set of metrics for all PPM problems, such that future research in the field will use this same set of metrics allowing for a direct comparsion between models.

4.2.1 Next activity & next attribute prediction

The work by Abb et al. [Abb et al., 2023] already describes some definitions for generalization in next activity prediction for PPM. This includes a model's ability to accurately predict the next activity for unseen activity sequences, for sequences with concurrency/loops and for sequences with attribute drift. Simple metrics like precision, recall, AUC, accuracy or F1-score can be used to quantify the performance on each of these scenarios.

A problem in PPM is the relatively high level of label ambiguity in event logs when considering only the activity prefix. Abb et al. have shown that state-of-the-art models are bounded by an accuracy limit caused by decision points in the process. It is unclear whether this problem can be overcome by creating more complex models, or whether this is an unavoidable problem. One way to get more insight into this, would be to leverage prediction probabilities to measure the model's behaviour in these ambiguous situations. Below we introduce a metric called the probability ratio score (PRS), which can be used to quantify how well a model is able to predict a label in such ambiguous situations. For each label j, r_j is the ratio of:

- The mean probability of j on samples where j is not the true label (incorrect).
- The mean probability of j on samples where j is the true label (correct).

thus:

$$r_j = \frac{\sum_i p_{ij} \left(1 - y_{ij}\right) \ / \ (N - N_j)}{\sum_i p_{ij} \left(y_{ij}\right) \ / \ N_j}$$

where:

- N: Number of test instances.
- N_j: Number of test instances with label j.
- $p_{i,j}$: Predicted probability of label j for test instance i.
- $y_{i,j} \in \{0, 1\}$: Indicates whether sample i has true label j.

To avoid r_j blowing up for instances where the mean probability of j on samples where j is the true label is extremely low, a log-sigmoid transform is applied resulting in the final form R_j:

$$R_j = \sigma\left(\ln(r_j + \epsilon)\right) = \frac{(r_j + \epsilon)}{1 + (r_j + \epsilon)}.$$

Here ϵ is a small constant set at 1e$-$7 to avoid taking ln(0) in some cases.
The final score PRS is given as the mean of R_j across all labels.
The resulting metric can be interpreted as follows:

- When $PRS \approx 0.5$: On average, the probability for labels on incorrect samples is similar to that on correct samples.
- When $PRS < 0.5$: On average, the model tends to give less probability to labels on wrong samples than on correct samples.
- When $PRS > 0.5$: On average, the model is giving higher probability to labels on incorrect samples than on correct ones.

4.2.2 Next timestamp prediction & remaining time prediction

A model trained to perform timestamp prediction should be able to predict the timestamp of the next activity (or remaining time) by leveraging various process attributes. A naive baseline would use a simple statistic like the mean or median time between activities to predict the next timestamp. A real business process likely has many factors influencing the duration in-between activities such as: resource availability, process dependencies, prioritization, resource experience and various external factors.

A model capable of generalization should be able to learn at least some of the time-influencing factors, and avoid generic predictions approaching a constant. It could thus be useful to compare the distributions of the predictions and actual targets to measure generalization. A correlation test can be used to compare the two distributions. A combination of correlation together with the RMSE, MSE or MAE, can be used to determine how well a model adjusts its predictions to the case execution context.

4.2.3 Activity suffix & attribute suffix prediction

Suffix prediction is a task where a model aims to predict the remaining sequence of attributes or activities for a running case. This type of prediction should be evaluated using the same methods as regular activity/attribute prediction, only using a metric suitable for sequences such as the Damerau-Levenshtein distance.

4.2.4 Outcome prediction

Outcome prediction can either be a regression or classification task. The goal here is to predict some outcome variable given an unfinished case in the same format as an event log. This type of prediction task is more similar to a classical machine learning problem compared to the tasks described above. As we are still dealing with process data, there might still be a risk of overfitting on the activity prefix, due to its the high levels of data leakage. For this reason the same methods as in next activity/attribute prediction will be used. It might also be necessary to balance the evaluation metrics for potentially unbalanced classes in the dataset.

4.2.5 Comparing generalization between models

Combined with appropriately debiased datasets, all metrics described in this section should give an indication on how well a PPM model will perform on unobserved data.

However, these metrics should not be used as a performance metric on their own, as different process circumstances are not captured by these metrics. The benchmark framework should not only incorporate these metrics into its evaluation functions, but also incorporate these metrics into various graphs that visualize the performance across different scenarios within a process.

4.3 Benchmark Framework

An initial version of the benchmark framework was implemented by following the requirements specified in section 3.3.1. The core idea behind the framework is to make it easy for researchers to run a standardized benchmark on their PPM models, while also enabling users to extend the benchmark with new datasets and metrics. Figure 4.2 shows the class diagram of the implemented framework. To reduce clutter, some of the implemented subclasses have been omitted. These are some of the subclasses extended from the abstract classes.

The framework consists of two core components: the BenchmarkLoader and Experiment classes. The BenchmarkLoader is responsible for initializing a benchmark, either by loading an already initialized benchmark from disk or by initializing a new one from a configuration file. The configuration can be used to specify which datasets should be used, how different benchmark tasks should be generated from these datasets, and which metrics should be used for evaluation.

When initializing a new benchmark from a configuration file, 4 submodules are used: DatasetLoaders, DatasetNormalizers, Metrics, and TaskGenerators. These submodules contain classes extended from their respective base model. These modules are designed in such a way that they can be easily extended for implementing new benchmarks. This section will proceed with a description of these submodules.

4.3.1 DatasetLoaders

The DatasetLoader classes inside the DatasetLoaders module are responsible for retrieving the source data. Currently supported methods are remote from url or local using a file path. The remote from url component requires a data owner to be specified, which is displayed to the user when downloading the data. This way, datasets with unclear licenses can be downloaded from their source with appropriate attributions.

4.3.2 DatasetNormalizers

The DatasetNormalizer classes inside the DatasetNormalizers module are responsible for formatting the data into a pandas dataframe, and use the XES naming conventions [xes, 2016]. Some important column names in this standard are as follows: the case identifier column should be named 'case:concept:name', the activity timestamp column should be named 'time:timestamp', the activity resource column should be named 'org:resource', and the activity column should be named 'concept:name'. DatasetNormalizers are also responsible for performing the debiased train/test split on a dataset, and calculating the target variable.

4.3.3 Metrics

Metric objects are wrappers around an evaluation function, and must contain an evaluate() method. Metrics are added to a task such that the evaluation method of a task returns the score on each metric.

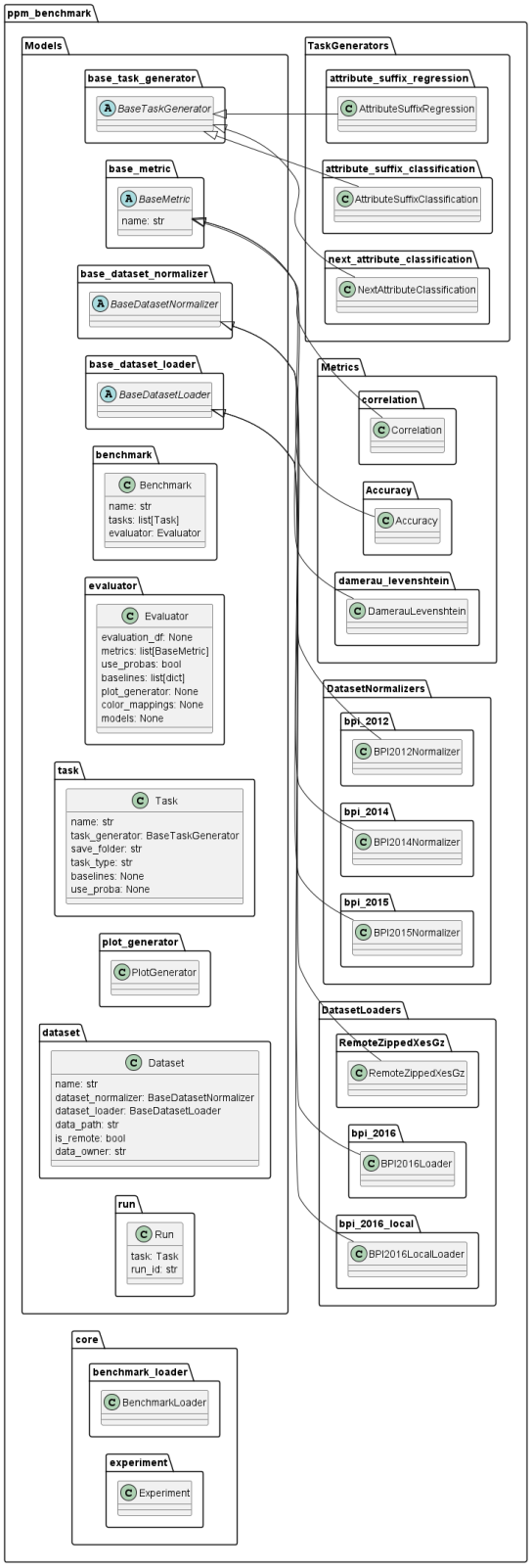
4.3.4 Configuration file

A configuration file is used to specify each of the submodules required for a benchmark, an example of such a configuration file can be found in appendix 1. Datasets are specified with their appropriate DatasetLoader, DatasetNormalizer, and TaskGenerators. A BenchmarkLoader object is then responsible for orchestrating the Benchmark initialization. After initialization the Benchmark is saved locally for easy retrieval.

4.3.5 Experiments

The Experiment class can be used for easy tracking of predictive process monitoring experiments. the init_run() method returns a Callback object which can be used to track training metadata and model evaluations. Two types of callbacks are implemented: a GenericCallback and TFCallback. The GenericCallback allows for the tracking of any PPM experiments by manually calling its methods inside the training loop. The TFCallback can be added to a Tensorflow model for automated tracking. Tracking is done inside Run objects which can be exported as a dictionary for easy processing of experiment data. The save() method on the experiment object can be used to save an experiment and its runs to disk.

Figure 4.2: The class diagram of developed benchmark framework. It provides a simplified overview of the framework's architecture, by visualizing the implemented classes and their relationships.



4.3.6 TaskGenerator

A TaskGenerator is responsible for generating baseline predictions and evaluation metrics for given prediction problems. A TaskGenerator has been implemented for the following

prediction problems: next attribute prediction, outcome prediction, and attribute suffix prediction. Each of these prediction problems has a TaskGenerator for both classification and regression.

Each TaskGenerator calculates the values needed to create the plots described in the next section (4.3.7). This includes calculating the closest train activity sequence for each test instance (in terms of Damerau Levensthein distance), finding attribute drift in the test set, and calculating baseline predictions. It should be noted that the calculation of the closest train sequence is an estimation and not an exact value. More details are given in 4.3.6.

The following baselines were implemented in the benchmark:

- **Next attribute classification:** a distance baseline giving the probability of each possible next attribute for the closest train activity sequence. A naive baseline giving the probability for each possible next attribute value, given the current attribute value, considering the entire training set.
- **Next attribute regression:** The median of all target values.
- **Outcome classification:** The mean probability for each class over the whole train set.
- **Outcome regression:** The median of all target values.
- **Attribute suffix classification:** The most common following attribute for the current attribute.

Activity sequence matching
A sequence matching algorithm was implemented that is used by both the next attribute classification distance baseline and one of the benchmark plot methods. This matching algorithm attempts to find the activity sequence in the training data closest to a given activity sequence, in terms of the Damerau-Levenshtein distance. The algorithm works by calculating the distance between a given sequence and a randomly selected sequence from the training data for each iteration. The algorithm returns the best matching sequence after no sequence with a lower distance is found after 100 iterations. This random search approach was chosen because of the computational cost of calculating the Damerau-Levenshtein distance for each combination of activity sequences.

4.3.7 Evaluator

The evaluator class is a wrapper around a PlotGenerator object, which is responsible for generating consistent graphs to evaluate a trained model. Each task has its own plotting methods that can be used to gain insight into the generalisation abilities of PPM models. Currently, three different plot methods have been implemented.

Figure 4.3 shows an example of a graph displaying the accuracy of a next activity prediction model by the distance from the train sequence. The model is a Random Forest (RF) classifier with default parameters. The plot is generated by calculating the Damerau-Levensthein distance between each test instance and the closest train sequence. The test instances are then grouped by this distance and predictions are made for each group. This plot shows the accuracy for each group, but any metric implemented in the benchmark is supported.

Figure 4.3: Example of a train sequence distance plot generated by the benchmark framework.

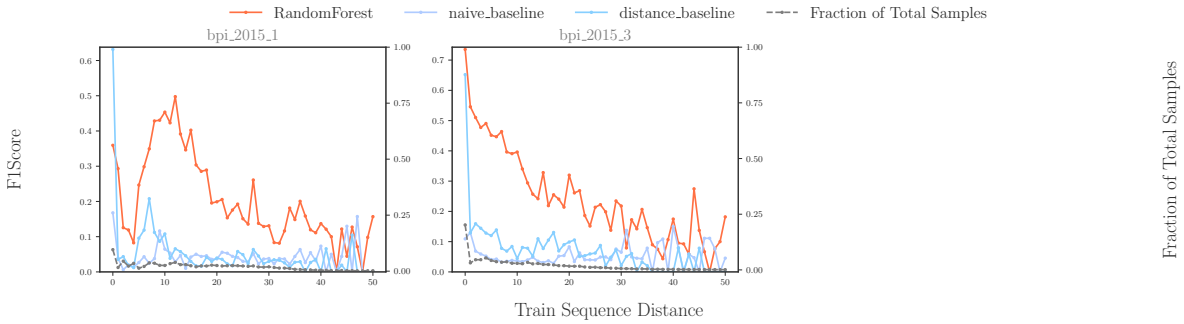
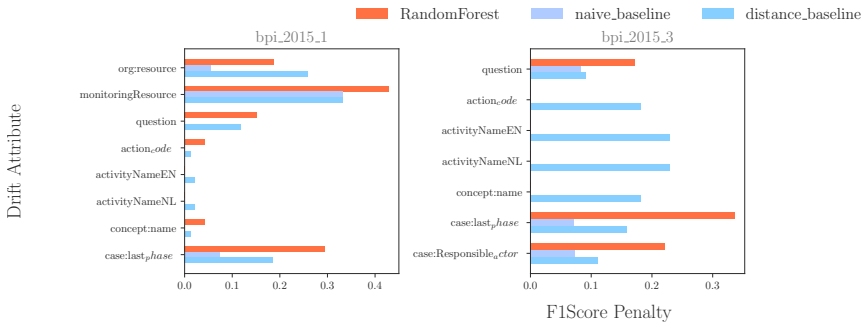


Figure 4.4: Example of an attribute drift plot generated by the benchmark framework.

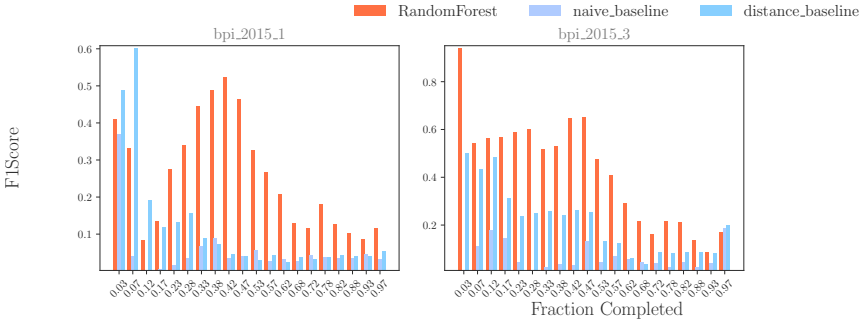


The motivation behind this graph lies in the observation made by Abb et al. [Abb et al., 2023] that most real-world event logs contain a large amount of example leakage between the train and test set. This graph aims to give an indication of how well a model is able to generalise by also measuring its performance on activity sequences not observed in the training data. If a model overfits on the activity prefix, this should be clearly visible in this plot.

Figure 4.4 shows the decrease in F1-score for test instances where some attribute value is not present in the training data. As with the previous plot, F1-score is displayed here, but any metric implemented in the benchmark is supported. This plot can be used to see how well a model can generalise for cases where attribute drift should have no impact on predictive performance. An example could be attribute drift in the org:resource column, where hiring a new employee should have no impact on performance. It gives a visual representation of the performance on the "attribute drift" generalisation task described in Section .

Figure 4.5 shows the accuracy of a model when grouped by its fraction completed. These fractions completed are binned in 20 groups. This graph is mostly useful for comparing the earliness of outcome predictions but could also be useful for finding individual strengths and weaknesses of models.

Figure 4.5: Example of a fraction completed plot generated by the benchmark framework.



Chapter 5

Experiments & Results

This chapter will describe the experiments that were performed to evaluate the benchmark framework. This includes a brief overview of the available literature to motivate the selection of models which will be evaluated on the benchmark. Section 5.2 contains the results of the experiments performed.

5.1 Experiment Setup

5.1.1 PPM literature

A literature review has been performed according to the method described in 3.5.1. The snowball method identified a total of 111 papers. Some of these papers can only be accessed through IEEE, for which Leiden University has no open access and therefore cannot be used for this research. Of the 111 articles, 32 have not included a reference to their source code, which disqualifies these articles for further analysis.

In total, 27 articles have published their source code and are accessible through Leiden University. Out of these 27 articles, 15 articles propose a method for next activity/attribute prediction, 2 articles propose a method for next timestamp prediction, 10 articles propose a method for remaining time prediction, 5 articles propose a method for outcome prediction, and 1 method was found for suffix prediction. Note that some methods can be used for multiple prediction types. Table 5.1 shows a list of the identified relevant papers.

Table 5.1: A table summarizing the results of the literature review. This table contains all identified relevant articles which are accessible (without paywall) and have their source code published. The title of the article is given together with a brief explanation of its relevance to this research.

Title	Reference	Relevance
Enhancing the Accuracy of Predictors of Activity Sequences of Business Processes	[Ali et al., 2023]	Proposes a new method for activity suffix and remaining time prediction. The method balances exploration and exploitation for predicting the next activity, which could have interesting implications for predictions on uncommon activity sequences.
ProcessTransformer: Predictive Business Process Monitoring with Transformer Network	[Bukhsh et al., 2021]	Authors propose a deep learning architecture leveraging transformers to predict next activities, remaining time and next event times. They claim an increase in performance compared to other methods on all problems on nearly all datasets.
Transition-driven time prediction for business processes with cycles	[Cao et al., 2022]	The authors propose a new method for remaining time prediction, they claim superior performance with respect to all compared methods.

Continued on next page

Title	Reference	Relevance
Remaining cycle time prediction with Graph Neural Networks for Predictive Process Monitoring	[Duong et al., 2023]	The authors propose a new method for remaining time prediction leveraging graph neural networks (GGNN). Results are measured on the helpdesk, BPIC 2020 and a non-public dataset. The method shows only a slight increase in performance compared to LSTM on BPIC 2020 vs helpdesk. The authors claim superiour performance on the non-public dataset.
PGTNet: A Process Graph Transformer Network for Remaining Time Prediction of Business Process Instances	[Elyasi et al., 2024]	This paper proposes a new method for remaining time prediction using a graph transformer network architecture. Authors claim superior performance on all compared datasets & methods. These methods include the previously mentioned ProcessTransformer and GGNN.
Semi-Supervised Discovery of DNN-Based Outcome Predictors from Scarcely-Labeled Process Logs	[Folino et al., 2022]	The authors propose a DNN-based outcome prediction model especially geared towards sparsely labelled outcome logs. They show an increase in performance compared to similar methods for outcome prediction.
Process data properties matter: Introducing gated convolutional neural networks (GCNN) and key-value-predict attention networks (KVP) for next event prediction with deep learning	[Heinrich et al., 2021]	This paper proposes two new neural network architectures for next activity prediction. The authors claim a significant performance increase as compared to more classical LSTM-based network architectures.
Exploiting Event Log Event Attributes in RNN Based Prediction	[Hinkka et al., 2019]	The authors propose a method which leverages an RNN-based architecture combined with a clustering approach which enables the use of event attributes as features for next activity prediction. The authors show that their approach of including event attributes can improve prediction accuracy.
Activity Prediction of Business Process Instances with Inception CNN Models	[Mauro et al., 2019]	The authors compare their newly proposed CNN-based model to an LSTM model. Results show that the CNN-based model outperforms LSTM-based models in terms of efficiency and accuracy. It should be noted that only 3 datasets were used for this comparison.
Time Matters: Time-Aware LSTMs for Predictive Business Process Monitoring	[Nguyen et al., 2020]	The authors propose a new 'T-LSTM' architecture for next activity and next timestamp prediction. They show an increased performance for 2 out of 2 datasets as compared to a more standard LSTM architecture.
Predicting remaining execution time of business process instances via auto-encoded transition system	[Ni et al., 2022]	The authors propose a new model for remaning time prediction leveraging an auto-encoded transition system. The approach shows superior performance as compared to other methods on the datasets hospital_billing, helpdesk and BPIC_2013. It does however not show superior performance on the BPIC_2012 dataset.
A Multi-View Deep Learning Approach for Predictive Business Process Monitoring	[Pasquadibisceglie et al., 2021b]	The authors propose another novel method for next activity prediction, levering multi-view and deep learning. The authors show that their proposed method outperforms all other methods on an extensive selection of datasets.
DARWIN: An online deep learning approach to handle concept drifts in predictive process monitoring	[Pasquadibisceglie et al., 2023]	The method proposed in this article aims to increase prediction accuracy by adjusting deep learning models for concept drift. The method outperforms most compared tree-based and ensemble-based models.
ORANGE: Outcome-Oriented Predictive Process Monitoring Based on Image Encoding and CNNs	[Pasquadibisceglie et al., 2020]	This paper proposes another CNN-based model for outcome prediction. Results show an increased performance as compared to SVM, LR, RF, XGB and LSTM based models.
FOX: a neuro-Fuzzy model for process Outcome prediction and eXplanation	[Pasquadibisceglie et al., 2021a]	The method proposed in this paper focuses on explainability in outcome predictions. Results show that the proposed method has a good trade-off between predictive accuracy and explainability, even out-performing the previously described ORANGE model in some cases.
Bayesian Network Based Predictions of Business Processes	[Pauwels et al., 2020]	This paper proposes a Bayesian network based model for next activity prediction. Results show that the proposed method is on-par with LSTM-based models and even outperforms these in some cases. This comes with the added benefit of explainability when using bayesian networks.
Incremental Predictive Process Monitoring: The Next Activity Case	[Pauwels et al., 2021]	The paper proposes a method for handling concept drift using an incremental learning strategy for next activity prediction. Results show that the method significantly improves 'no-update' results without having to fully retrain the model. Interestingly, this is especially true for BPIC_2015 datasets which have an accuracy of 0.2-0.3 without update, and 0.72-0.76 using the proposed method.

Continued on next page

Title	Reference	Relevance
A Deep Adversarial Model for Suffix and Remaining Time Prediction of Event Sequences	[Taymouri et al., 2021]	In this paper, a deep adversarial model is proposed for both activity suffix and remaining time prediction. The authors show their method outperforms all compared methods on 4 datasets, for both remaining time and activity suffix in terms of SDL.
Predictive Business Process Monitoring via Generative Adversarial Nets: The Case of Next Event Prediction	[Taymouri et al., 2020]	This papers proposes a similar adversarial network as the paper described above. The authors here show that their method outperforms all other compared methods in terms of weighted average accuracy on the same 4 datasets as the paper above. An interesting observation here is that both this paper and the paper above compare their method to [Tax et al., 2017], and the paper above includes this paper in the comparison as well. However, this paper reports the performance in terms of weighted average accuracy, while the other paper reports performance in terms of SDL. This makes it impossible to compare both methods.
Improving Predictive Process Monitoring Through Reachability Graph-Based Masking of Neural Networks	[Theis et al., 2023]	This paper describes another novel approach for next activity prediction which leverages reachability graphs to mask the neural networks of decay replay mining methods. Results show in increased performance as compared to regular decay replay mining methods in terms of AUROC.
Decay Replay Mining to Predict Next Process Events	[Theis et al., 2019]	Decay Replay Mining leverages petri net models enchanced with time decay functions to create continuous process state samples. these samples are used in combination with discrete token movement counters and Petri net markings to train a deep learning model for next event prediction. The method outperforms all other compared methods on 9 datasets.
Enhancing Stochastic Petri Net-based Remaining Time Prediction using k-Nearest Neighbors	[Vandenabeele et al., 2022]	In this article, a method is proposed for remaining time prediction which combines petri-nets with k-nearest neighbors. Unfortunately, the authors do not compare their method to state-of-the-art methods.
Encoding High-Level Control-Flow Construct Information for Process Outcome Prediction	[Vazifehdoostirani et al., 2022]	The method proposed in this paper aims to encode high-level process features into the prediction model for outcome prediction. Results show that the approach outperforms vanilla LSTM models on a large selection of datasets.
Predicting process performance: A white-box approach based on process models	[Verenich et al., 2019]	This paper proposes a method which is used to perform remaining time prediction using explainable techniques. These techniques include an analysis of reachability and computing the mean cycle time for the reachable activities. On average, the proposed method outperforms baselines such as an XGBoost model or stochastic petri net. No comparison was made to more state-of-the-art methods.
Predicting Outcomes of Business Process Executions Based on LSTM Neural Networks and Attention Mechanism	[Wang et al., 2021]	This paper describes an outcome prediction method leveraging and LSTM with an attention mechanism. The proposed method outperforms all other compared methods.
XNAP: Making LSTM-based Next Activity Predictions Explainable by Using LRP	[Weinzierl et al., 2020]	This paper proposes a LSTM based architecture integrated with a layer-wise relevance propagation method, to allow for better explainability. Performance is similar to other LSTM based methods.
Learning Uncertainty with Artificial Neural Networks for Improved Remaining Time Prediction of Business Processes	[Weytjens et al., 2021]	Although this paper describes a novel approach for remaining time prediction, this approach is based around predicting distributions over specific values.

5.1.2 Model selection

Due to time constraints it is unfeasible to benchmark all models described in table 4.1. This section will motivate how the selection of models was made.

One goal of this evaluation is to see if the proposed benchmark can give a better estimation of a model's real-world performance as compared to popular methods described in literature. Another goal is to create a unified benchmark which allows for a direct comparison of models by using the same datasets, with the same pre-processing, on the same performance metrics.

A good selection of models with which to evaluate the framework should therefore include: models which have similar architectures but different reported evaluation metrics,

models with very different architectures (to estimate the influence of event log bias on reported performance), and models with limited experimental results (to validate claims over more datasets). The following articles were selected based on these criteria:

Next activity prediction:

- **Graph-based masking** [Theis et al., 2023]: included as it provides a drastically different approach compared to more standard NN-based methods.
- **ProcessTransformer** [Bukhsh et al., 2021]: included as the state-of-the-art method.
- **Bayesian network based** [Pauwels et al., 2020]: included as it is a very simple method, it would be interesting to see differences with state-of-the-art methods.

Remaining time prediction:

- **PGTNet** [Elyasi et al., 2024]: will serve as the state-of-the-art model with highest reported performance.
- **ProcessTransformer** [Bukhsh et al., 2021]: included to compare against PGTNet and validate claims of PGTNet paper.
- **Petri-net + KNN** [Vandenabeele et al., 2022]: included as is it much simpler than the other two methods and it would be interesting to compare the performance.

Outcome prediction:

- **Control-flow features** [Vazifehdoostirani et al., 2022]: Included to see the effects of high-level control-flow features on the debiased datasets.
- **LSTM + Attention mechanism** [Wang et al., 2021]: will serve as the state-of-the-art model.

5.1.3 Final setup

To summarize, an initial version of the benchmark framework was implemented which will be used to measure the predictive performance of 8 models described in 5.1.2. The performance will be measured on the debiased datasets listed in table 4.1. By using the evaluator from the benchmark framework, the predictive performance will be compared using three plots:

- Train sequence distance: A plot that shows the performance of a model on the test set, where test set prefixes are grouped by their distance from the closest train prefix. This could give insight into how well a model performs on test prefixes not observed during training. It should give insight in both the “unseen activity sequences” and “label ambiguity (in terms of the activity prefix)” tasks described in 4.3.7.
- Attribute drift: A plot that shows the performance penalty for test instances for which some attribute value was not observed during training. It should give insight in the ”attribute drift” task described in 4.3.7.

- Fraction completed: A plot that shows the performance on the test set grouped by the fraction completed of each test instance. This plot is not directly meant to answer research questions, but could give interesting insights into early prediction quality and differences in models.

For next activity predictions (or next attribute classification) tasks the PRS metric (4.2.1) was added to the framework. This metric might help in measuring how well a model "recognizes" test prefixes with multiple theoretically valid predictions. It assigns a value close to 0 if the model assigns a higher probability to labels even if they are misclassified.

Furthermore, several baseline predictions were added by the benchmark framework for each task to see how well a model performs compared to some naive predictions.

5.2 Results

The following section contains the results from the performed experiments. As this research followed a design science methodology, this section will describe the results grouped by their corresponding cycle or objective.

5.2.1 Relevance cycle

Even though subsection 5.1.2 has described a selection of 10 models across 4 prediction types, the final evaluation was done on only 4 models. This was due to unexpected difficulties in implementing the chosen algorithms despite the source code being available. This made it too time consuming to implement all algorithms. The decision could have been made to simply adjust the methodology subsection, however, I think that the experienced difficulties underline why this research is relevant. This subsection will contain a description of these difficulties as I think they can be considered relevant 'results'.

Project dependencies

An issue encountered with nearly every algorithm is the poorly documented dependencies. Even if a project contained a requirements file, this commonly did not include any package versions. This made it very time consuming to find compatible packages especially for projects using a deep learning framework like Tensorflow or pyTorch. This is because setting up the appropriate CUDA versions can be an error-prone and time consuming process when having to try different versions. Pm4py is another package which is used by nearly all algorithms and which caused a lot of issues. The package has undergone many major changes in its API over the years, requiring you to find a very specific version of the package. Documentation on earlier versions of the package is also hard to find, making it difficult to make code adjustments.

Data Pre-processing

Most projects are published/designed in such a way that it runs the complete pre-processing and training pipeline for only the datasets used in the article. This is likely because source code is only published to reproduce results and not implement a model for custom data. However, the goal of this research is to benchmark the models on the debiased datasets. These datasets are different from the regular BPIC datasets commonly used in the published code. Thus, the pre-processing code for each algorithm had to be adjusted to work with the debiased data and developed benchmark tool. This was by far the most time consuming task. Algorithms could run just fine after adjusting some

pre-processing steps, but then produce strange results on the benchmark after the (usually days long) training process had finished. This was usually due to an oversight in the pre-processing code like a hidden test-set shuffling mechanism, or faulty parameter setup hidden in some configuration file.

Petri-net generation
Some of the algorithms described in subsection 5.1.2 required a petri net as input. Petri nets are usually generated using pm4py, which as mentioned earlier, was a difficult library to work with. I was not able to generate a petri net for all datasets. When pm4py did not work, I tried using ProM [ProM Tools, 2025] but with no success. The exact cause is not directly clear. It might have to do with the mining algorithm not being able to produce a petri-net due to the structure of the event log. Interestingly, datasets for which I was not able to generate a petri-net were also not present in the original article.

5.2.2 Knowledge base

In the end four models were benchmarked on the developed framework: ProcessTransformer next activity & remaining time [Bukhsh et al., 2021], PGTNet [Elyasi et al., 2024] and PyDREAM (NAP) [Theis et al., 2019]. For both PGTNet and PyDREAM, I was unable to train a model on all datasets. The evaluation graphs seen in this subsection contain only the joined results, the full results for ProcessTransformer can be found in Appendix 2.

5.2.3 Next activity prediction

Table 5.2 contains the results for next activity prediction for the ProcessTransformer and NAP models. It also shows the predicted results from two baseline methods. The naive baseline simply returns the most common next activity given the current activity, while the distance baseline returns the most common next activity based on the closest training prefix. The reported F1-score is added for datasets where it is available in the original article. One thing to note about this table is that the metrics are calculated on all test instances where both models made a prediction. For the ProcessTransformer model, this means that there are some slight differences with the table in appendix 2. This is because NAP requires at least three activities to make a prediction.

The results for the F1-score and PRS metric are summarized in figures 5.2 and 5.1. From the F1-scores we can see that overall, the naive baseline performs worst, while the distance baseline performance is very similar to that of the state-of-the-art models. The bpi_2020_domestic dataset seems to be a special case where all performance metrics are identical.

The results measured using the PRS metric seem to correlate with the F1-score performance. Meaning PRS is generally lower where the F1-score is higher. The ProcessTransformer model seems to be the only model that outperforms the distance baseline in terms of PRS. This might suggest that this model is learning more complex relations in the process. This is supported by the observation that ProcessTranformer is the best performing model overall in terms of F1-score.

The results show that the reported F1-score of the ProcessTransformer model is slightly higher compared to the score observed on most of the debiased datasets. For the NAP model the reported F1-score on the traffic_fine is equal to the benchmark score while it is substantially lower on the bpi_2012 dataset. For the helpdesk dataset, both

Figure 5.1: Comparison of F1-score across eight event logs. Each subplot compares the F1-score for four approaches: ProcessTransformer (orange), NAP (green), a naive baseline (light blue), and a distance baseline (darker blue). Higher F1-score values signify better predictive performance.

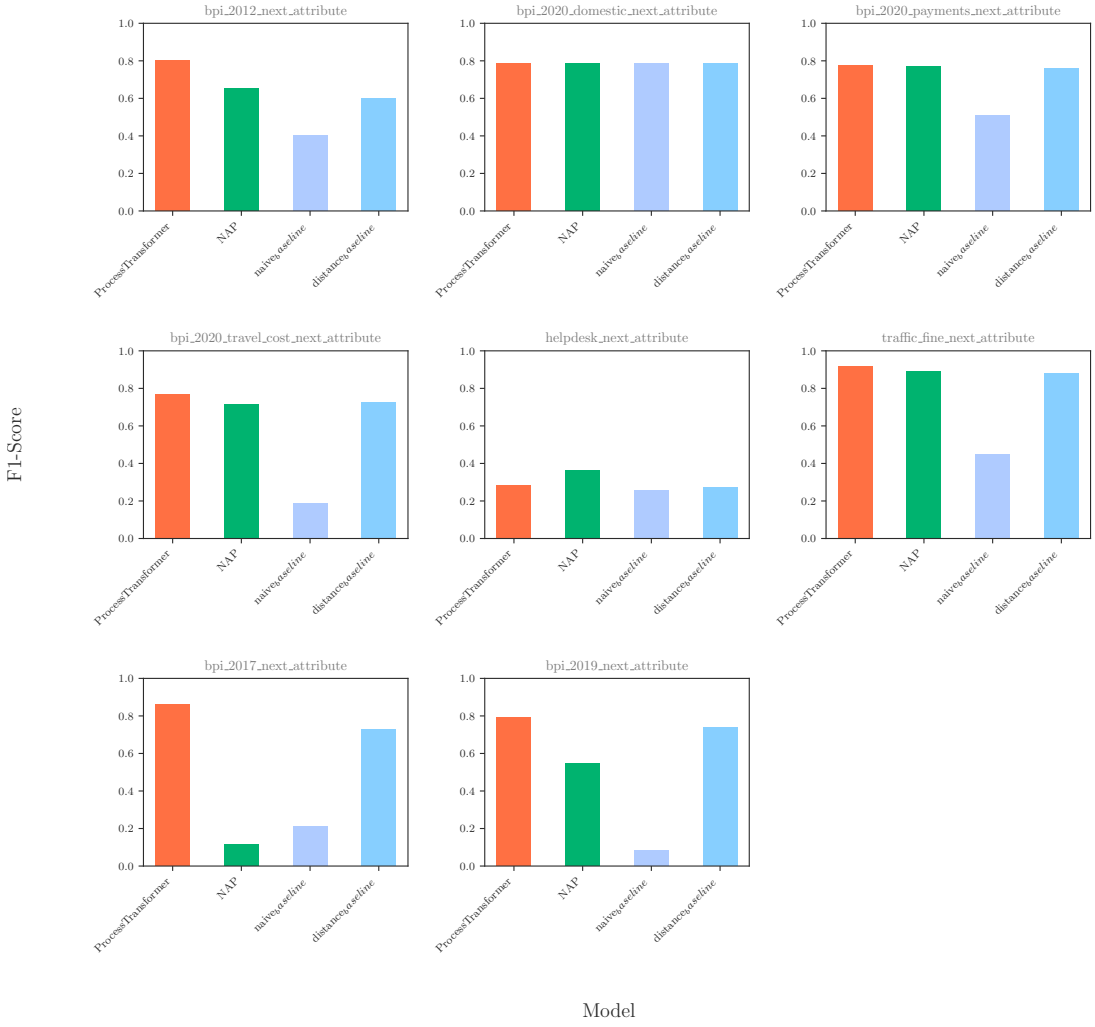
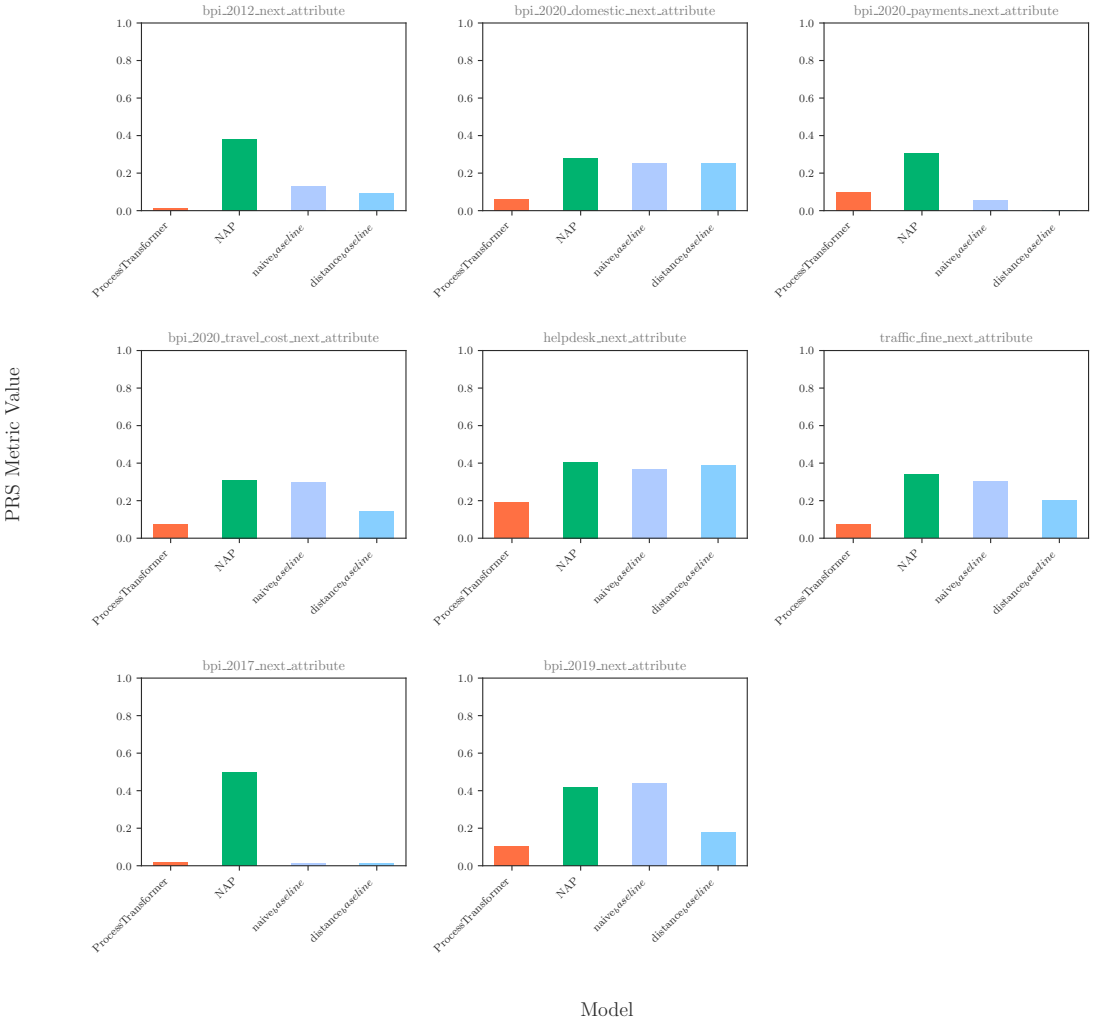


Figure 5.2: Comparison of PRS across eight event logs. Each subplot compares the PRS for four approaches: ProcessTransformer (orange), NAP (green), a naive baseline (light blue), and a distance baseline (darker blue). Lower PRS values signify better predictive performance.



models perform significantly worse on the debiased version. Both models have a reported F1-score of approximately 0.8, and a benchmark score of approximately 0.3 on this dataset. When looking at the ProcessTransformer predictions in Appendix 2, this score is slightly higher at around 0.437 but still much lower compared to the reported score.

ProcessTransformer clearly outperforms all other models in most datasets. However, it should be noted that in only two out of eight datasets, the ProcessTransformer model is ahead by more than 6% in terms of accuracy when compared to the distance baseline. NAP only outperforms the other methods in terms of accuracy for the helpdesk dataset.

Except for the helpdesk dataset, the NAP model does not appear to achieve better performance than the distance baseline. Especially the bpi_2017 dataset seems to be a challenge for this model, with only an accuracy of 0.123 it performs even worse than the naive baseline.

Interestingly, the naive baseline performs poorly on nearly all datasets except for the bpi_2020_domestic dataset. This might indicate that the debiasing strategy has worked, as this is the opposite of what was reported in the article by Abb et al. [Abb et al., 2023].

Table 5.2: A comparative overview of prediction scores (Accuracy, PRS, Precision, Recall) for ProcessTransformer, NAP, a naive baseline, and distance baseline models across multiple datasets. Higher values of Accuracy, Precision, Recall indicate better performance, while the perfect score for PRS is 0 and PRS $\geq$ 0.5 is undesirable. The final metric, "Reported F1-Score," shows F1-Score values from the original articles (where available).

Dataset	Metric	ProcessTransformer	NAP	Naive baseline	Distance baseline
bpi_2012	Accuracy	**0.836**	0.625	0.451	0.629
bpi_2012	PRS	**0.015**	0.382	0.129	0.094
bpi_2012	Precision	**0.800**	0.745	0.429	0.613
bpi_2012	Recall	**0.836**	0.625	0.451	0.629
bpi_2012	F1-Score	**0.802**	0.653	0.403	0.602
bpi_2012	Reported F1-Score	**0.83**	**0.763**	–	–
bpi_2020_domestic	Accuracy	**0.850**	0.847	**0.850**	0.843
bpi_2020_domestic	PRS	**0.063**	0.280	0.254	0.251
bpi_2020_domestic	Precision	0.745	0.741	0.742	**0.817**
bpi_2020_domestic	Recall	**0.850**	0.847	**0.850**	0.843
bpi_2020_domestic	F1-Score	**0.787**	0.785	**0.787**	0.785
bpi_2020_domestic	Reported F1-Score	**0.861**	–	–	–
bpi_2020_payments	Accuracy	**0.843**	0.836	0.596	0.819
bpi_2020_payments	PRS	0.097	0.307	0.056	**0.005**
bpi_2020_payments	Precision	0.803	0.725	0.480	**0.826**
bpi_2020_payments	Recall	**0.843**	0.836	0.596	0.819
bpi_2020_payments	F1-Score	**0.778**	0.771	0.508	0.758
bpi_2020_payments	Reported F1-Score	–	–	–	–
bpi_2020_travel_cost	Accuracy	**0.798**	0.766	0.243	0.756
bpi_2020_travel_cost	PRS	**0.072**	0.310	0.300	0.146
bpi_2020_travel_cost	Precision	**0.796**	0.688	0.176	0.746
bpi_2020_travel_cost	Recall	**0.798**	0.766	0.243	0.756
bpi_2020_travel_cost	F1-Score	**0.769**	0.715	0.190	0.728
bpi_2020_travel_cost	Reported F1-Score	–	–	–	–
helpdesk	Accuracy	0.325	**0.438**	0.285	0.299
helpdesk	PRS	0.189	**0.406**	0.369	0.389
helpdesk	Precision	0.268	0.317	0.252	**0.348**
helpdesk	Recall	0.325	**0.438**	0.285	0.299
helpdesk	F1-Score	0.284	**0.366**	0.256	0.272
helpdesk	Reported F1-Score	**0.82**	**0.795**	–	–
traffic_fine	Accuracy	**0.929**	0.877	0.454	0.869
traffic_fine	PRS	**0.072**	0.339	0.301	0.200
traffic_fine	Precision	**0.929**	0.906	0.445	0.906
traffic_fine	Recall	**0.929**	0.877	0.454	0.869
traffic_fine	F1-Score	**0.917**	0.890	0.448	0.881
traffic_fine	Reported F1-Score	0.87	**0.89**	–	–
bpi_2017	Accuracy	**0.876**	0.123	0.244	0.738
bpi_2017	PRS	0.016	0.499	**0.013**	0.014
bpi_2017	Precision	**0.864**	0.111	0.332	0.735
bpi_2017	Recall	**0.876**	0.123	0.244	0.738

Continued on next page

Dataset	Metric	ProcessTransformer	NAP	Naive baseline	Distance baseline
bpi_2017	F1-Score	**0.859**	0.116	0.212	0.727
bpi_2017	Reported F1-Score	–	–	–	–
bpi_2019	Accuracy	**0.775**	0.546	0.087	0.719
bpi_2019	PRS	**0.105**	0.418	0.440	0.177
bpi_2019	Precision	**0.838**	0.591	0.078	0.793
bpi_2019	Recall	**0.775**	0.546	0.087	0.719
bpi_2019	F1-Score	**0.793**	0.547	0.081	0.740
bpi_2019	Reported F1-Score	–	–	–	–

Train sequence distance

Figure 5.3 shows the F1-score for each model on the test set, grouped by the test prefix's distance from the closest prefix in the training data. The results indicate that it might be beneficial to perform such analysis compared to simply displaying a results table. The results table 5.2 suggested that the NAP model significantly outperformed the ProcessTransformer model on the helpdesk dataset. However, when looking at the results in this graph, it appears that the NAP model only achieves this higher performance because it outperforms ProcessTransformer on prefixes which were already observed during training and which consist of a significant portion of the total test samples. When accounting for this, the results might be considered less impressive.

The plot offers several other notable insights. When looking at the bpi_2012 dataset for example, an F1-score of 0.653 for the NAP model might sound decent, but the plot clearly shows this score is much lower for prefixes which were not already present in the training data. This clearly indicates that the model is not very capable of generalisation, even though its overall score might sound decent. The bpi_2020_domestic and bpi_2020_payments plots also raise some questions. The two compared models only score marginally better compared to the baselines for some samples, but have identical scores for most. The identical scores are quite strange and might have to do with some of the unique properties of these datasets.

Furthermore, when looking at the bpi_2020_travel_cost dataset, the NAP model and distance baseline have very similar scores in the results table (0.715 vs 0.728) with the distance baseline even outperforming NAP. However, after looking at the graph, one might conclude that NAP clearly outperforms the distance baseline. The overall F1-score being slightly higher for the baseline is only caused by the fact that it is only slightly better for samples with distance 0, which account for a much larger portion of test samples thus inflating the overall score.

Fraction completed

Figure 5.4 displays the F1-score for each model on the test set, grouped by the test prefix's completion percentage at the time of prediction. This plot gives another unique perspective of the performance of a model under various conditions. For the helpdesk dataset, it is once again visible that the NAP model only outperforms the others on very specific samples. It would be interesting to investigate what these samples have in common. For the bpi_2020_domestic and bpi_2020_payments datasets, the identical results are even more pronounced. The traffic_fine dataset now also shows some identical results, which were not directly visible from the plot earlier.

Figure 5.3: Comparison of predictive performance across eight event logs grouped by the test prefix distance from the closest train prefix. Each subplot compares F1-score for four approaches—ProcessTransformer (orange), NAP (green), a naive baseline (light blue), and a distance baseline (darker blue)—as the training sequence distance increases (x-axis). The dashed gray line in each subplot indicates the fraction of total samples at each distance (right-hand y-axis). Higher F1-scores values signify better predictive performance.

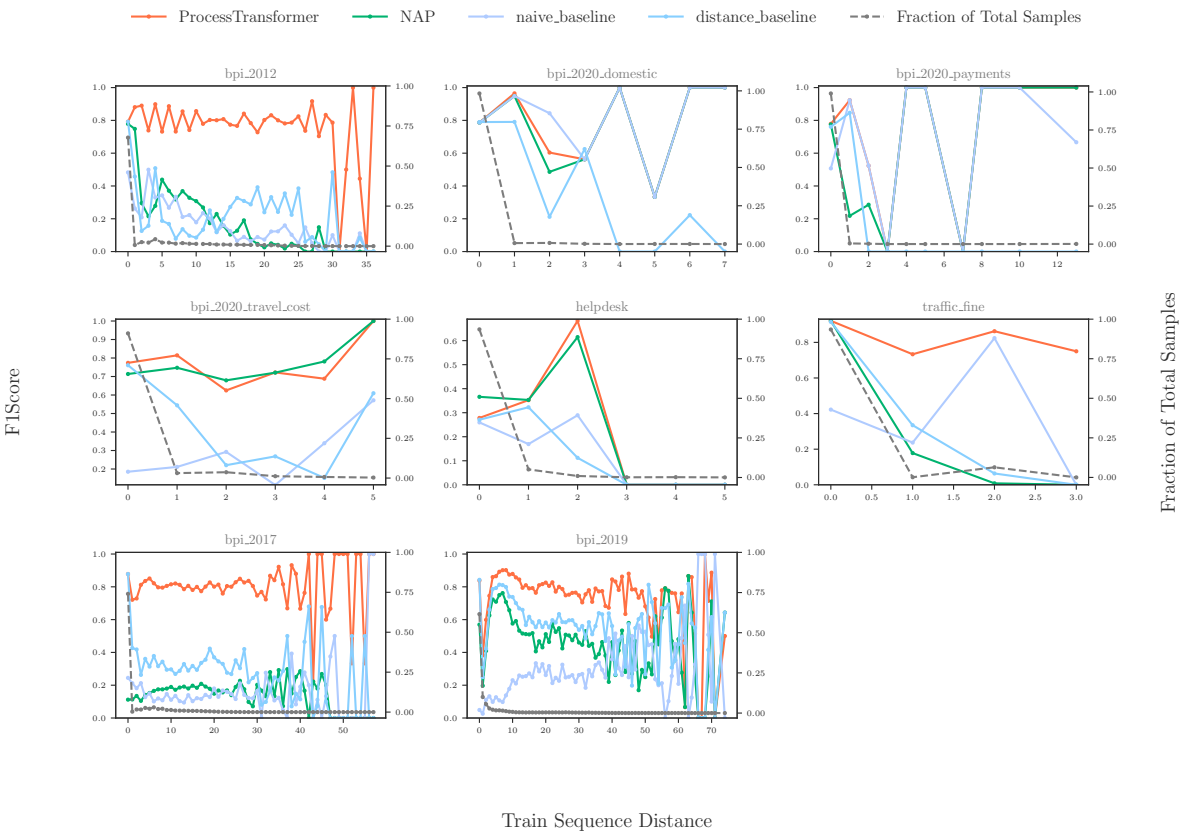


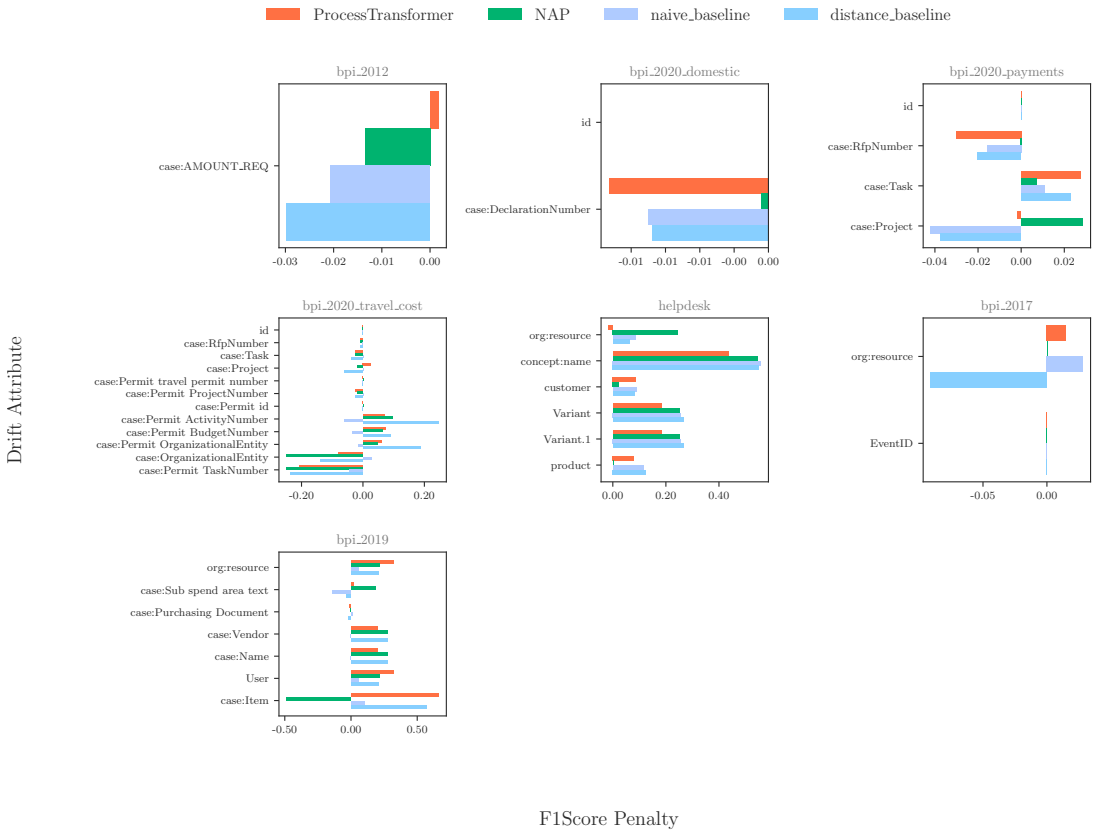
Figure 5.4: Comparison of predictive performance across eight event logs grouped by the fraction complete of the test prefix. Each subplot compares F1-score for four approaches—ProcessTransformer (orange), NAP (green), a naive baseline (light blue), and a distance baseline (darker blue)-as the fraction completed increases (x-axis). Higher F1-scores signify better predictive performance.

Attribute drift

Figure 5.4 shows the average reduction in F1-score for test instances where the displayed attribute's value (y-axis) was not in the training data. The plot provides some interesting insights into how well a model can handle a changing process. The helpdesk dataset stands out as it has a significant penalty for the 'concept:name' attribute, which contains the name of the activity. It is not very surprising that a model built to predict activities has a reduction in performance when new activities are added. Appendix 2 also contains plots from the ProcessTransformer on the bpi_2015 datasets, which also contain attribute drift in the concept:name column. Here we see a similar trend, but at 0.05 - 0.2 the penalty does seem lower for these datasets. An important column to look for in this plot is the "org:resource" column; this contains the resource that performed the activity. As noted in the problem statement, under ideal circumstances, a model should not take a significant hit in performance if a new resource is added to the process. In the plot, we see that the performance impact of new resources is only substantial on the bpi_2019 and bpi_2020_travel_cost datasets.

The bpi_2020_travel_cost, bpi_2019 and helpdesk datasets seem to be the only datasets for which there is a significant impact on performance for cases with attribute drift. It seems like these are also the datasets on which models have the worst predictive performance. This trend continues in table 2 and figure 1 (appendix 2). Although it makes sense that when plotting errors, the datasets with the highest errors are the ones with the worst predictive performance, it might point in the direction of where miss-

Figure 5.5: Comparison of the average reduction in F1-score across eight event logs for test instances where some attribute value was not observed in the training data (attribute drift). Each subplot compares the average decrease in F1-score (F1-score penalty) for four approaches: ProcessTransformer (orange), NAP (green), a naive baseline (light blue), and a distance baseline (darker blue). The y-axis indicates the attribute, the x-axis shows the average F1-score penalty. Lower F1-score penalty values signify a model is more capable of dealing with attribute drift for the given attribute.



classifications come from. When we look at the bpi_2017 dataset, the performance impact of the attribute drift columns is negligible, but the model does not have a perfect F1-score (0.859). Meaning on average, the impact on performance is the same as if there was no attribute drift. Combining this with the PRS metric value which is near perfect for bpi_2017, it might indicate that dataset/model suffers from the accuracy limit problem described in Subsection 4.2.1. However, for the helpdesk and bpi_2019 datasets, there seems to be room for improvement in terms of PRS and some attribute drift columns have a significant impact on performance. There might be specific unique behaviours in cases where attribute drift occurs for these datasets which the model was not able to capture, and thus does not generalise well on.

Remaining time prediction

Table 5.3 shows the results of the remaining time prediction benchmark on the ProcessTransformer and PGTNet model. The included baseline is simply a constant, it returns the median remaning time across the training set.

The results immediately show that the remaining time scores take a significant hit on the debiased datasets. The ProcessTransformer model performs worse on all data sets compared to the baseline. With a correlation of near 0 on all datasets, it seems like this model is not able to predict the remaining time at all. The PGTNet model can only outperform the baseline in some bpi_2015 datasets.

Table 5.3: A comparative overview of prediction errors (MAE, RMSE, MSE) and correlation for ProcessTransformer, PGTNet, and a baseline model across multiple datasets. Lower values of MAE, RMSE, and MSE indicate better performance, while the opposite is true for correlation. The final metric, "Reported MAE," shows MAE values from the original articles (where available).

Dataset	Metric	ProcessTransformer	PGTNet	Baseline
bpi_2012	MAE	8.378	9.205	**7.919**
bpi_2012	RMSE	10.835	12.555	**10.714**
bpi_2012	MSE	117.399	157.634	**114.781**
bpi_2012	Correlation	**0.062**	0.034	0.000
bpi_2012	Reported MAE	4.60	**2.31**	–
bpi_2015_1	MAE	44.901	**28.793**	39.364
bpi_2015_1	RMSE	67.157	**49.700**	61.553
bpi_2015_1	MSE	4510.016	**2470.076**	3788.775
bpi_2015_1	Correlation	-0.101	**0.621**	0.000
bpi_2015_1	Reported MAE	–	–	–
bpi_2015_2	MAE	64.218	**46.778**	55.249
bpi_2015_2	RMSE	89.580	**77.968**	80.264
bpi_2015_2	MSE	8024.543	**6079.068**	6442.347
bpi_2015_2	Correlation	0.049	**0.404**	0.000
bpi_2015_2	Reported MAE	–	–	–
bpi_2015_3	MAE	20.762	**11.973**	14.382
bpi_2015_3	RMSE	32.211	**20.838**	26.608
bpi_2015_3	MSE	1037.522	**434.204**	708.006
bpi_2015_3	Correlation	-0.060	**0.601**	0.000
bpi_2015_3	Reported MAE	–	–	–
bpi_2015_4	MAE	58.072	50.483	**41.644**
bpi_2015_4	RMSE	70.446	63.830	**51.982**
bpi_2015_4	MSE	4962.596	4074.292	**2702.176**
bpi_2015_4	Correlation	-0.015	**0.408**	0.000
bpi_2015_4	Reported MAE	–	–	–
bpi_2020_domestic	MAE	5.500	5.453	**4.845**
bpi_2020_domestic	RMSE	11.925	11.857	**11.669**
bpi_2020_domestic	MSE	142.208	140.597	**136.154**
bpi_2020_domestic	Correlation	**0.004**	-0.008	0.000
bpi_2020_domestic	Reported MAE	2.44	**1.19**	–
bpi_2015_5	MAE	43.725	**41.361**	43.265
bpi_2015_5	RMSE	64.531	64.044	**55.003**
bpi_2015_5	MSE	4164.191	4101.611	**3025.350**
bpi_2015_5	Correlation	0.083	**0.221**	0.000
bpi_2015_5	Reported MAE	–	–	–

Train sequence distance Figure 5.6 shows the train sequence distance plot for the remaining time prediction results shown above. As with next activity prediction, the plot provides some valuable insights into model behaviour which cannot be deduced from performance metrics alone. the bpi_2015 datasets, it appears that the test sets contain a large amount of cases which differ significantly from the training data. This explains the poor performance of the ProcessTransformer model, which uses a feature set that relies heavily on the activity prefix. This probably also explains the poor performance of ProcessTransformer on the bpi_2015 datasets for the next activity prediction (Appendix 2). Interestingly, these are the datasets in which PGTNet achieves a significantly better performance compared to the baseline. In fact, it seems like the higher the train sequence distance, the better the performance. Unfortunately, the plots generated by the

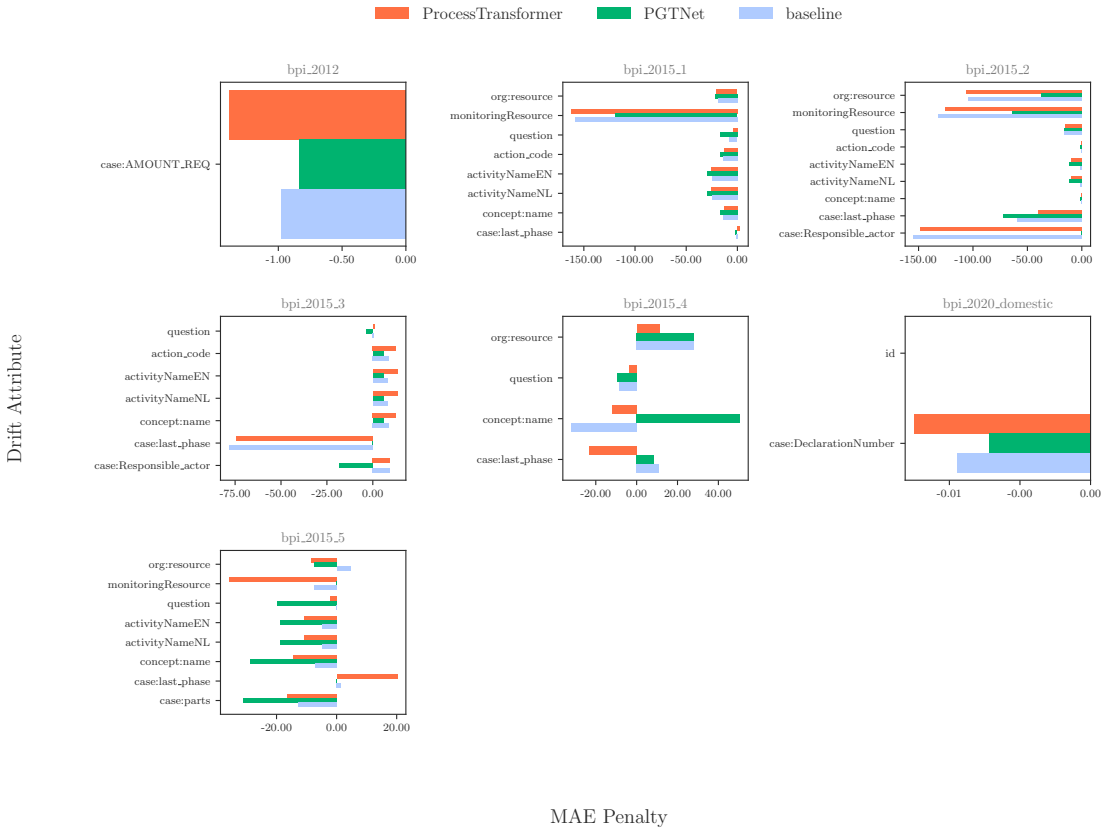
Figure 5.6: Comparison of predictive performance across seven BPIC datasets grouped by the test prefix distance from the closest train prefix. Each subplot compares mean absolute error (MAE) for three approaches—ProcessTransformer (orange), PGTNet (green), and a baseline model (blue)—as the training sequence distance increases (x-axis). The dashed gray line in each subplot indicates the fraction of total samples at each distance (right-hand y-axis). Lower MAE values signify better predictive performance.

benchmark are not detailed enough to explain this behaviour.

Attribute drift The attribute drift plot seen in figure 5.7 reveals another unique perspective on model behaviour, which cannot be deduced from the performance metrics alone. This is especially true for the bpi_2015 dataset. Here, we see that the resource or monitoring resource has a significant influence on the MAE score. The fact that this is also the case for the baseline indicates that this is not an overfitting problem, but more likely has to do with a complete shift in the process. Interestingly, the PGTNet model seems to handle this shift much better compared to the other models. It would be interesting to investigate exactly what happens here, as it might also explain the observed performance in the next activity prediction on these datasets for the ProcessTransformer model.

Since the performance penalty for attribute drift on the bpi_2012 and bpi_2020_domestic dataset seems to be negligible, while the overall performance is not better than an extremely naive baseline, suggests that some properties of these datasets prevent the algorithms from learning the remaining time. It is not directly clear from the results why this happens and is something that needs to be further investigated.

Figure 5.7: Comparison of the average reduction in mean absolute error score (MAE Penalty) across seven BPIC datasets for test instances where some attribute value was not observed in the training data (attribute drift). Each subplot compares the average decrease in MAE (MAE penalty) for three approaches: ProcessTransformer (orange), PGTNet (green), and a baseline model (blue). The y-axis indicates the attribute, the x-axis shows the average MAE penalty. Higher MAE penalty values signify a model is more capable of dealing with attribute drift for the given attribute.



5.2.4 Design cycle

Benchmark plots

The results in this chapter clearly show the benefit of the developed benchmark framework over a more standard approach to only compare performance metrics. We have seen in the bpi_2020_travel_cost dataset for next activity prediction that the F1-score indicates that the baseline outperforms NAP, while the train sequence distance plot tells a different story. A similar thing happened for the bpi_2012 dataset, for which the NAP model reported a decent overall score, but closely looking at the plot shows us that this score is only thanks to a large portion of prefixes which were already observed during training. In the remaining time prediction problem, the same plot shows that a likely culprit of poor performance is the large number of cases which are completely different from the training data. It also shows that even though the performance is not very good, PGTNet is able to capture at least some relationship between case properties and the remaining time, serving as a good starting point for further investigations.

The fraction completed plot has proven useful for visualizing unexpected prediction behaviors in next activity prediction. Some data sets show strange identical performance on some groups. It is difficult to determine why this happens, but explaining this phenomenon could help to improve the benchmark and possibly also the prediction models.

The attribute drift plot has made a clear distinction between data sets that have reduced performance because of concept/attribute drift, and datasets for which attribute drift is not the cause of decreased performance. However, it appears that this is only the case for the remaining time prediction. ProcessTransformer seems to run perfectly fine on those same datasets in next activity prediction. In fact, we might even observe the accuracy limit problem for the ProcessTransformer model on these datasets when looking at the PRS metric. This shows us that the proposed metric could be useful for scoring models on these types of data set.

Dataset debiasing

The debiasing strategy also looks promising from the results of the experiments. In next activity prediction, the naive baseline is clearly not able to make accurate predictions. This is completely opposite to the results reported by Abb et al. [Abb et al., 2023], who observed a naive baseline that was on par with the state-of-the-art models in biased datasets. Furthermore, in remaining time prediction, the reported MAE was significantly better than the observed MAE on the debiased datasets. Combined with the fact that no model provides truly impressive performance in this problem, suggests that published methods for remaining time prediction rely heavily on bias in the datasets.

Framework flaws

There were also some flaws in the benchmark framework that were noticed during the evaluation. An oversight in the design is the fact that models can have different requirements for test instances on which to make predictions. For example, the PGTNet model uses a graph-based neural network which requires at least a trace of length 3. This means that the first two rows of each test case are removed from the data. For the results in this chapter, these types of test set mismatches were handled manually, but a more robust approach should be implemented for this. There is also a similar problem where some algorithms cannot handle unseen attribute values while others can. This could result in somewhat unfair comparisons, as some models have made much fewer predictions.

Chapter 6

Discussion

This chapter will delve into the interpretation of the experimental results of the previous chapter and analyse the implications for predictive process monitoring. This chapter will also address the limitations of this research and provide suggestions for future research.

6.1 Limitations

Before we discuss the conclusions of this work, we need to address some of its limitations. The results observed in chapter 4 were the result of a somewhat limited experiment. First, only three unique algorithms were implemented compared to the 111 articles identified during the literature review. This means that a large portion of the algorithms were not compared. Conclusions about the meaning of these results on the general state of PPM might become invalid once more models are benchmarked.

Secondly, there exist some limitations in the experiment setup itself which might result in somewhat unfair comparisons. Models were only trained once, no hyperparameter optimisation was done on the debiased datasets (the best reported hyper parameters were used), and no statistical analysis was done for model comparison.

Finally, a reflection on the benchmark framework itself will be inherently subjective. More objective validation will be possible as the framework is adopted and extended by other researchers.

6.2 Interpretation of results

6.2.1 Debiasing strategy

The debiasing strategy used to create the datasets has proven to be a very useful tool to measure generalisability. The results show a reduction in performance compared to the reported performance. This is especially true for remaining time prediction. This likely means that the reported performance was based on some bias contained in the datasets. The debiasing strategy leaves a slightly biased part of the dataset near the splitting point, which has not been corrected yet. There is also a problem where some datasets become completely unusable due to a large number of test cases being removed. To advance the field, more high-quality (large) event logs are needed to produce an even better benchmark.

6.2.2 Measuring generalisation

The experimental results have clearly shown that trying to measure the performance of PPM models using only standard machine learning evaluation metrics and standard datasets will not produce reliable estimates of a model's real-world performance. Performance scores appear to be heavily inflated by data leakage and, without the context of a baseline's performance, give no information on generalisation as the actual 'difficulty' of the prediction problem cannot be assessed. The results show several examples of models with high performance scores that are very similar to the baseline score. This is especially true for remaining-time prediction, where the naive baseline even outperforms the compared models multiple times.

The plots generated by the benchmark framework appear to be a great tool to gain more understanding of generalisation. The train sequence distance plot can help identify performance scores inflated by data leakage, as well as help identify models that are capable of generalisation. Examples of this are the bpi_2012 and traffic_fine subplots for next activity prediction. The fraction completed plot provides a more granular insight into performance at specific moments in the process. It is not directly useful as a tool to measure generalisation, but can serve as a starting point for more detailed analysis on model behaviour. It might be useful to investigate the identical performances observed in these plots. The attribute drift plot aids in understanding generalisation by giving insight into the performance penalty suffered by models when some process attribute changes. As with the previous plot, it could be used as a starting point for more detailed analysis.

6.2.3 PRS metric

The introduced PRS metric has proven to be useful in identifying data sets for which the accuracy limit described by Abb et al. might be reached, but also shows data sets that still have potential for improved scores. Examples of this are the bpi_2017 and bpi_2019 datasets for which the observed performance clearly outperforms the baseline, but the bpi_2019 has a much higher PRS indicating potential room for improvement. However, on its own, this metric might not provide much information. A more complete and detailed comparison between models is needed to evaluate this metric's usefulness.

6.3 Answers to the research questions

RQ1: *Which methods can be further developed which can effectively measure the generalisation ability of predictive process monitoring models across diverse generalisation scenarios?* The results of this research show that generalisation cannot be reduced to a single metric, and instead a combination of quantitative and visual analysis is needed to assess the generalisability of PPM models. The combination of baseline comparisons, the newly proposed PRS metric combined with more standard machine learning metrics, and detailed visualisations provide the information necessary to assess generalisation.

RQ2: *What models can be found in PPM literature for each of the prediction tasks found in literature?* The literature review uncovered 28 methods with published source code. Ultimately, four models from three distinct studies were implemented.

RQ3: *What are the software requirements for a benchmark suite which allows for easy integration of the methods designed in RQ 1, and allows for the addition of new methods in the future?* Before starting the development of the benchmark framework, a

set of requirements was established which can be found in Section 3.3.1. These include requirements for an API that can maintain consistency even when adding new metrics, datasets, or evaluation metrics. A 'skeleton' version of the benchmark was implemented, extended with new datasets and metrics, and then tested by applying it to the experiments described in Chapter 5.

The developed skeleton framework has proven to be sufficient to build an extendable benchmark framework that allows for easy addition of new datasets and evaluation metrics. This conclusion is supported by the fact that no changes were made to the skeleton framework when implementing the various datasets and metrics used in this research.

However, one design flaw was encountered during the evaluation of models on the benchmark. The design of the framework assumes that all models are able to make a prediction on all test set instances; however, this is not the case in practice. For the models evaluated in this research, some require a longer activity prefix in order to make a prediction than others. There are possibly also other restrictions which were not encountered now, but might become an issue later. For now, this problem has been solved by adding a method to remove specific test cases from the test data. This is not an ideal solution as it requires figuring out which instances should be removed manually. Furthermore, it puts some models at a disadvantage in direct comparisons as it might remove correct predictions from the results because another model is not able to make these predictions. Future iterations of the framework should implement a more robust approach to handle this issue.

RQ4: *What is the performance of the identified state-of-the-art models (RQ2) on the constructed benchmark, and how does this compare to the performance described by the authors?* The benchmark results indicate a significant drop in performance, especially for remaining time prediction, compared to the published results. This difference suggests that previously reported scores are inflated by the bias present in the datasets.

6.4 Future research

The most obvious starting point for future research would be to benchmark more models. Unfortunately, there was not enough time available to do that during this thesis. Both the suffix and outcome prediction problems remain untouched. It would be interesting to see the benchmark results for these model types. Benchmarking more remaining time and next activity models could also be helpful in gaining a more accurate overview on the current state of PPM.

Future work should also focus on extending the benchmark with more advanced analysis options. The attribute drift and fraction completed plots have shown that predictive performance can be identical for the baseline and compared models for some subsets of the event log. Understanding why this happens could help to advance the field significantly. Future versions of the benchmark should also implement a better approach for handling instances where some models are not able to make a prediction on all test cases.

Finally, more fundamental work could be done to define more accurate definitions with respect to what generalisation really means in the context of PPM. The discussion written by Abb et al. [Abb et al., 2023], which was referenced throughout this thesis, provides some great starting points for this. These points were briefly discussed in Section 2.4.1, but the true spirit of this discussion was not captured in the final benchmark.

Chapter 7

Conclusion

For this master's thesis, a benchmark framework was implemented with the goal of providing more insight into the performance of predictive process monitoring (PPM) techniques. Recent publications in the field have raised some questions with regard to the validity of reported performance of PPM models, as event logs appear to contain large amounts of bias. Abb et al. [Abb et al., 2023] showed that a naive baseline performs on par with state-of-the-art models in next activity prediction. Weytjens and de Weerdt [Weytjens and De Weerdt, 2021] made a similar point and have shown that popular event logs used for PPM benchmarking contain large amounts of data leakage. The authors proposed a debiasing strategy which was used as the foundation for the developed benchmark. Other publications in the field further emphasise the need for a common and reliable benchmark, as current publications all use different evaluation metrics, data pre-processing techniques, and data sets [Teinemaa et al., 2019], [Rama-Maneiro et al., 2021], [Kratsch et al., 2021]. The developed benchmark contains features for debiasing event logs, experiment tracking, baseline prediction generation, and detailed performance analysis. A literature review was conducted to identify state-of-the-art models for various PPM prediction problems. Finally, four of these models were implemented and tested on the developed framework.

7.1 Final Reflections

The findings of this thesis underscore the complexity of accurately estimating the real world performance of PPM models. The results have shown that high levels of bias are present in unprocessed event logs that cause inflated performance scores. It has become clear that, in their current state, PPM models are not sufficiently reliable to be used in the real world. Future work should focus on gaining a better understanding of model behaviour and using this understanding to improve models. All in all, the developed benchmark proves to be a valuable contribution to the field of PPM, and I encourage all researchers to contribute to the project[1]and use this framework to benchmark their models.

[1]https://github.com/hiddevr/ppm_benchmark

Bibliography

[xes, 2016] (2016). Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50.

[Abb et al., 2023] Abb, L., Pfeiffer, P., Fettke, P., and Rehse, J.-R. (2023). A discussion on generalization in next-activity prediction. In *International Conference on Business Process Management*, pages 18–30. Springer.

[Ali et al., 2023] Ali, M. A. et al. (2023). Enhancing the accuracy of predictors of activity sequences of business processes. *arXiv.org*.

[Bukhsh et al., 2021] Bukhsh, Z. A. et al. (2021). Processtransformer: Predictive business process monitoring with transformer network. *arXiv: Learning*.

[Cao et al., 2022] Cao, R. et al. (2022). Transition-driven time prediction for business processes with cycles. *Expert Systems with Applications*.

[de Leoni and Mannhardt, 2015] de Leoni, M. and Mannhardt, F. (2015). Road Traffic Fine Management Process.

[Dees and van Dongen, 2016] Dees, M. and van Dongen, B. B. (2016). BPI Challenge 2016.

[Duong et al., 2023] Duong, L. T. et al. (2023). Remaining cycle time prediction with graph neural networks for predictive process monitoring. In *International Conference on Machine Learning Technologies*.

[Elyasi et al., 2024] Elyasi, K. A. et al. (2024). Pgtnet: A process graph transformer network for remaining time prediction of business process instances. In *International Conference on Advanced Information Systems Engineering*.

[Folino et al., 2022] Folino, F. et al. (2022). Semi-supervised discovery of dnn-based outcome predictors from scarcely-labeled process logs. *Business & Information Systems Engineering*.

[Heinrich et al., 2021] Heinrich, K. et al. (2021). Process data properties matter: Introducing gated convolutional neural networks (gcnn) and key-value-predict attention networks (kvp) for next event prediction with deep learning. *Decision Support Systems*.

[Hevner, 2007] Hevner, A. R. (2007). The three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19(2):87.

[Hinkka et al., 2019] Hinkka, M. et al. (2019). Exploiting event log event attributes in rnn based prediction. In *Communications in computer and information science*.

[Kratsch et al., 2021] Kratsch, W., Manderscheid, J., Röglinger, M., and Seyfried, J. (2021). Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. *Business & Information Systems Engineering*, 63:261–276.

[Levy, 2014] Levy, D. (2014). Production Analysis with Process Mining Technology.

[Mannhardt, 2016] Mannhardt, F. (2016). Sepsis cases - event log.

[Mannhardt, 2017] Mannhardt, F. (2017). Hospital billing - event log.

[Mauro et al., 2019] Mauro, N. D. et al. (2019). Activity prediction of business process instances with inception cnn models. In *International Conference of the Italian Association for Artificial Intelligence*.

[Nguyen et al., 2020] Nguyen, A. et al. (2020). Time matters: Time-aware lstms for predictive business process monitoring. In *Lecture Notes in Computer Science*.

[Ni et al., 2022] Ni, W. et al. (2022). Predicting remaining execution time of business process instances via auto-encoded transition system. *Intelligent Data Analysis*.

[Pasquadibisceglie et al., 2020] Pasquadibisceglie, V. et al. (2020). Orange: Outcome-oriented predictive process monitoring based on image encoding and cnns. *IEEE Access*.

[Pasquadibisceglie et al., 2021a] Pasquadibisceglie, V. et al. (2021a). Fox: a neuro-fuzzy model for process outcome prediction and explanation. In *International Conference on Process Mining*.

[Pasquadibisceglie et al., 2021b] Pasquadibisceglie, V. et al. (2021b). A multi-view deep learning approach for predictive business process monitoring. *IEEE Transactions on Services Computing*.

[Pasquadibisceglie et al., 2023] Pasquadibisceglie, V. et al. (2023). Darwin: An online deep learning approach to handle concept drifts in predictive process monitoring. *Engineering applications of artificial intelligence*.

[Pauwels et al., 2020] Pauwels, S. et al. (2020). Bayesian network based predictions of business processes. In *Lecture Notes in Business Information Processing*.

[Pauwels et al., 2021] Pauwels, S. et al. (2021). Incremental predictive process monitoring: The next activity case. In *Lecture Notes in Computer Science*.

[Peffers et al., 2020] Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., and Bragge, J. (2020). Design science research process: A model for producing and presenting information systems research. *arXiv preprint arXiv:2006.02763*.

[Polato, 2017] Polato, M. (2017). Dataset belonging to the help desk log of an Italian Company.

[ProM Tools, 2025] ProM Tools (2025). Prom tools. Accessed: 2025-01-06.

[Rama-Maneiro et al., 2021] Rama-Maneiro, E., Vidal, J. C., and Lama, M. (2021). Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Transactions on Services Computing*, 16(1):739–756.

[Scheid et al., 2018] Scheid, M., Rehse, J.-R., Houy, C., and Fettke, P. (2018). Data Set for MobIS Challenge 2019.

[Steeman, 2014] Steeman, W. (2014). BPI Challenge 2013.

[Tax et al., 2017] Tax, N., Verenich, I., La Rosa, M., and Dumas, M. (2017). Predictive business process monitoring with lstm neural networks. In *Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings 29*, pages 477–492. Springer.

[Taymouri et al., 2020] Taymouri, F. et al. (2020). Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In *International Conference on Business Process Management*.

[Taymouri et al., 2021] Taymouri, F. et al. (2021). A deep adversarial model for suffix and remaining time prediction of event sequences. In *SDM*.

[Teinemaa et al., 2019] Teinemaa, I., Dumas, M., Rosa, M. L., and Maggi, F. M. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(2):1–57.

[Theis et al., 2019] Theis, J. et al. (2019). Decay replay mining to predict next process events. *IEEE Access*.

[Theis et al., 2023] Theis, J. et al. (2023). Improving predictive process monitoring through reachability graph-based masking of neural networks. *IEEE Transactions on Computational Social Systems*.

[van der Aalst, 2022] van der Aalst, W. M. (2022). Process mining: a 360 degree overview. In *Process Mining Handbook*, pages 3–34. Springer.

[van Dongen, 2011] van Dongen, B. (2011). Real-life event logs - Hospital log.

[van Dongen, 2012] van Dongen, B. (2012). BPI Challenge 2012.

[van Dongen, 2017] van Dongen, B. (2017). BPI Challenge 2017.

[van Dongen, 2019] van Dongen, B. (2019). BPI Challenge 2019.

[van Dongen, 2020a] van Dongen, B. (2020a). BPI Challenge 2020.

[van Dongen, 2020b] van Dongen, B. (2020b). Bpi challenges: 10 years of real-life datasets.

[van Dongen, 2014] van Dongen, B. B. (2014). BPI Challenge 2014.

[van Dongen, 2015] van Dongen, B. B. (2015). BPI Challenge 2015.

[Vandenabeele et al., 2022] Vandenabeele, J. et al. (2022). Enhancing stochastic petri net-based remaining time prediction using k-nearest neighbors. *arXiv*.

[Vazifehdoostirani et al., 2022] Vazifehdoostirani, M. et al. (2022). Encoding high-level control-flow construct information for process outcome prediction. In *International Conference on Process Mining*.

[Verenich et al., 2019] Verenich, I. et al. (2019). Predicting process performance: A white-box approach based on process models. *J. Softw. Evol. Process.*

[Wang et al., 2021] Wang, J. et al. (2021). Predicting outcomes of business process executions based on lstm neural networks and attention mechanism. *RS*.

[Weinzierl et al., 2020] Weinzierl, S. et al. (2020). Xnap: Making lstm-based next activity predictions explainable by using lrp. In *Lecture Notes in Business Information Processing*.

[Weinzierl et al., 2024] Weinzierl, S., Zilker, S., Dunzer, S., and Matzner, M. (2024). Machine learning in business process management: A systematic literature review. *Expert Systems with Applications*, 253:124181.

[Weytjens and De Weerdt, 2021] Weytjens, H. and De Weerdt, J. (2021). Creating unbiased public benchmark datasets with data leakage prevention for predictive process monitoring. In *International Conference on Business Process Management*, pages 18–29. Springer.

[Weytjens et al., 2021] Weytjens, H. et al. (2021). Learning uncertainty with artificial neural networks for improved remaining time prediction of business processes. In *Lecture Notes in Computer Science*.

Appendix 1: Sample Benchmark Configuration

```
datasets:
  - name: "bpi_2012"
    dataset_normalizer: "BPI2012Normalizer"
    dataset_loader: "LocalXes"
    data_path: "../raw_eventlogs/BPI_Challenge_2012.xes/BPI_Challenge_2012.
    is_remote: false
    data_owner: "Boudewijn van Dongen"
    tasks:
      - name: "bpi_2012_next_attribute"
    split_details:
      start_date: null
      end_date: null

  - name: "bpi_2020_travel_cost"
    dataset_normalizer: "BPI2020Normalizer"
    dataset_loader: "LocalXes"
    data_path: "../raw_eventlogs/BPI_Challenge_2020_PrepaidTravelCost.xes/I
    is_remote: false
    data_owner: "Boudewijn van Dongen"
    tasks:
      - name: "bpi_2020_travel_cost_next_attribute"
    split_details:
      start_date: null
      end_date: null

benchmark:
    task_type: "next_attribute"
    name: "Test Benchmark"
    save_folder: "next_attribute_classification"
    attr_col: "concept:name"
    keywords_dict: null
    evaluator: "NextAttributeClassification"

    tasks:
      - name: "bpi_2012_next_attribute"
        save_folder: "next_attribute_classification/bpi_2012_next_attribute
        task_generator:
          name: "NextAttributeClassification"

      - name: "bpi_2020_travel_cost_next_attribute"
        save_folder: "next_attribute_classification/bpi_2020_travel_cost_ne
        task_generator:
          name: "NextAttributeClassification"

    metrics:
```

- name: "Accuracy"
- name: "LASS"
- name: "Precision"
- name: "Recall"
- name: "F1Score"

Appendix 2: ProcessTransformer results

Table 1: A comparative overview of prediction errors (MAE, RMSE, MSE) and correlation for ProcessTransformer and a baseline model across multiple datasets. Lower values of MAE, RMSE, and MSE indicate better performance, while the opposite is true for correlation. The final metric, "Reported MAE," shows MAE values from the original articles (where available).

Dataset	Metric	ProcessTransformer	Baseline
bpi_2012	MAE	8.392	7.868
bpi_2012	RMSE	10.876	10.663
bpi_2012	MSE	118.297	113.703
bpi_2012	Correlation	0.059	0.000
bpi_2012	**Reported MAE**	–	–
bpi_2015_1	MAE	46.600	40.910
bpi_2015_1	RMSE	69.307	63.593
bpi_2015_1	MSE	4803.482	4044.132
bpi_2015_1	Correlation	-0.108	0.000
bpi_2015_1	**Reported MAE**	–	–
bpi_2015_2	MAE	68.126	61.613
bpi_2015_2	RMSE	97.363	88.552
bpi_2015_2	MSE	9479.482	7841.392
bpi_2015_2	Correlation	0.031	0.000
bpi_2015_2	**Reported MAE**	–	–
bpi_2014	MAE	2.959	2.259
bpi_2014	RMSE	4.800	4.069
bpi_2014	MSE	23.038	16.561
bpi_2014	Correlation	0.003	0.000
bpi_2014	**Reported MAE**	–	–
bpi_2015_3	MAE	23.284	17.466
bpi_2015_3	RMSE	36.720	32.594
bpi_2015_3	MSE	1348.378	1062.367
bpi_2015_3	Correlation	-0.038	0.000
bpi_2015_3	**Reported MAE**	–	–
bpi_2015_4	MAE	59.984	44.262
bpi_2015_4	RMSE	73.068	54.569
bpi_2015_4	MSE	5338.891	2977.751
bpi_2015_4	Correlation	-0.039	0.000
bpi_2015_4	**Reported MAE**	–	–
bpi_2020_domestic	MAE	5.481	4.830
bpi_2020_domestic	RMSE	11.543	11.277
bpi_2020_domestic	MSE	133.240	127.160
bpi_2020_domestic	Correlation	0.008	0.000
bpi_2020_domestic	**Reported MAE**	–	–
bpi_2015_5	MAE	42.797	42.935
bpi_2015_5	RMSE	63.150	54.172
bpi_2015_5	MSE	3987.878	2934.586
bpi_2015_5	Correlation	0.092	0.000
bpi_2015_5	**Reported MAE**	–	–
bpi_2020_payments	MAE	6.759	6.071
bpi_2020_payments	RMSE	11.442	10.969
bpi_2020_payments	MSE	130.925	120.316
bpi_2020_payments	Correlation	-0.052	0.000
bpi_2020_payments	**Reported MAE**	–	–
bpi_2020_travel$_{c}ost$	MAE	18.643	16.323
bpi_2020_travel$_{c}ost$	RMSE	32.309	31.962
bpi_2020_travel$_{c}ost$	MSE	1043.871	1021.541
bpi_2020_travel$_{c}ost$	Correlation	-0.018	0.000
bpi_2020_travel$_{c}ost$	**Reported MAE**	–	–
helpdesk	MAE	18.775	14.410
helpdesk	RMSE	23.105	18.087
helpdesk	MSE	533.852	327.130
helpdesk	Correlation	-0.025	0.000
helpdesk	**Reported MAE**	**3.72**	–
traffic_fine	MAE	304.079	287.373
traffic_fine	RMSE	364.164	363.371
traffic_fine	MSE	132615.175	132038.212
traffic_fine	Correlation	-0.033	0.000
traffic_fine	**Reported MAE**	**98.24**	–
bpi_2017	MAE	11.855	10.340
bpi_2017	RMSE	15.442	14.005
bpi_2017	MSE	238.454	196.150
bpi_2017	Correlation	0.009	0.000
bpi_2017	**Reported MAE**	–	–

Table 2: A comparative overview of prediction scores (Accuracy, PRS, Precision, Recall) for ProcessTransformer, a naive baseline and distance baseline model across multiple datasets. Higher values of Accuracy, PRS, Precision, Recall indicate better performance, while the perfect score for PRS is 0 and PRS $\geq$ 0.5 is undesirable. The final metric, "Reported F1-Score," shows F1-Score values from the original articles (where available).

Dataset	Metric	ProcessTransformer	Naive Baseline	Distance Baseline
bpi_2012	Accuracy	0.843	0.487	0.641
bpi_2012	PRS	0.014	0.13	0.017
bpi_2012	Precision	0.805	0.464	0.621
bpi_2012	Recall	0.843	0.487	0.641
bpi_2012	F1Score	0.81	0.441	0.612
bpi_2012	**Reported F1-Score**	**0.83**	–	–
bpi_2015_1	Accuracy	0.24	0.054	0.113
bpi_2015_1	PRS	0.2	0.321	0.374
bpi_2015_1	Precision	0.235	0.065	0.133
bpi_2015_1	Recall	0.24	0.054	0.113
bpi_2015_1	F1Score	0.226	0.049	0.115
bpi_2015_1	**Reported F1-Score**	–	–	–
bpi_2015_3	Accuracy	0.404	0.065	0.217
bpi_2015_3	PRS	0.134	0.177	0.016
bpi_2015_3	Precision	0.428	0.071	0.223
bpi_2015_3	Recall	0.404	0.065	0.217
bpi_2015_3	F1Score	0.398	0.057	0.207
bpi_2015_3	**Reported F1-Score**	–	–	–
bpi_2020_domestic	Accuracy	0.867	0.868	0.85
bpi_2020_domestic	PRS	0.07	0.04	0.011
bpi_2020_domestic	Precision	0.796	0.767	0.826
bpi_2020_domestic	Recall	0.867	0.868	0.85
bpi_2020_domestic	F1Score	0.81	0.811	0.803
bpi_2020_domestic	**Reported F1-Score**	**0.861**	–	–
bpi_2020_payments	Accuracy	0.859	0.687	0.83
bpi_2020_payments	PRS	0.104	0.267	0.189
bpi_2020_payments	Precision	0.801	0.577	0.807
bpi_2020_payments	Recall	0.859	0.687	0.83
bpi_2020_payments	F1Score	0.798	0.611	0.772
bpi_2020_payments	**Reported F1-Score**	–	–	–
bpi_2015_5	Accuracy	0.442	0.024	0.256
bpi_2015_5	PRS	0.125	0.218	0.247
bpi_2015_5	Precision	0.468	0.04	0.255
bpi_2015_5	Recall	0.442	0.024	0.256
bpi_2015_5	F1Score	0.435	0.018	0.247
bpi_2015_5	**Reported F1-Score**	–	–	–
bpi_2020_travel_cost	Accuracy	0.821	0.324	0.771
bpi_2020_travel_cost	PRS	0.074	0.2	0.12
bpi_2020_travel_cost	Precision	0.82	0.267	0.762
bpi_2020_travel_cost	Recall	0.821	0.324	0.771
bpi_2020_travel_cost	F1Score	0.794	0.28	0.746
bpi_2020_travel_cost	**Reported F1-Score**	–	–	–
helpdesk	Accuracy	0.516	0.491	0.501
helpdesk	PRS	0.154	0.418	0.381
helpdesk	Precision	0.394	0.383	0.453
helpdesk	Recall	0.516	0.491	0.501
helpdesk	F1Score	0.437	0.418	0.432
helpdesk	**Reported F1-Score**	**0.82**	–	–
traffic_fine	Accuracy	0.948	0.6	0.863
traffic_fine	PRS	0.075	0.189	0.396
traffic_fine	Precision	0.949	0.59	0.891
traffic_fine	Recall	0.948	0.6	0.863
traffic_fine	F1Score	0.94	0.594	0.87
traffic_fine	**Reported F1-Score**	**0.87**	–	–
bpi_2017	Accuracy	0.862	0.258	0.728
bpi_2017	PRS	0.017	0.108	0.041
bpi_2017	Precision	0.853	0.342	0.731
bpi_2017	Recall	0.862	0.258	0.728
bpi_2017	F1Score	0.846	0.223	0.719
bpi_2017	**Reported F1-Score**	–	–	–
bpi_2019	Accuracy	0.667	0.066	0.626
bpi_2019	PRS	0.131	0.396	0.284
bpi_2019	Precision	0.73	0.074	0.696
bpi_2019	Recall	0.667	0.066	0.626
bpi_2019	F1Score	0.662	0.068	0.623
bpi_2019	**Reported F1-Score**	–	–	–

Figure 1: Comparison of the average reduction in F1-score penalty across ten event logs for test instances where some attribute value was not observed in the training data (attribute drift). Each subplot compares the average decrease in F1-score (F1-score penalty) for three approaches: ProcessTransformer (orange), a naive baseline (light blue), and a distance baseline (darker blue). The y-axis indicates the attribute, the x-axis shows the average F1-score penalty. Lower F1-score penalty values signify a model is more capable of dealing with attribute drift for the given attribute.

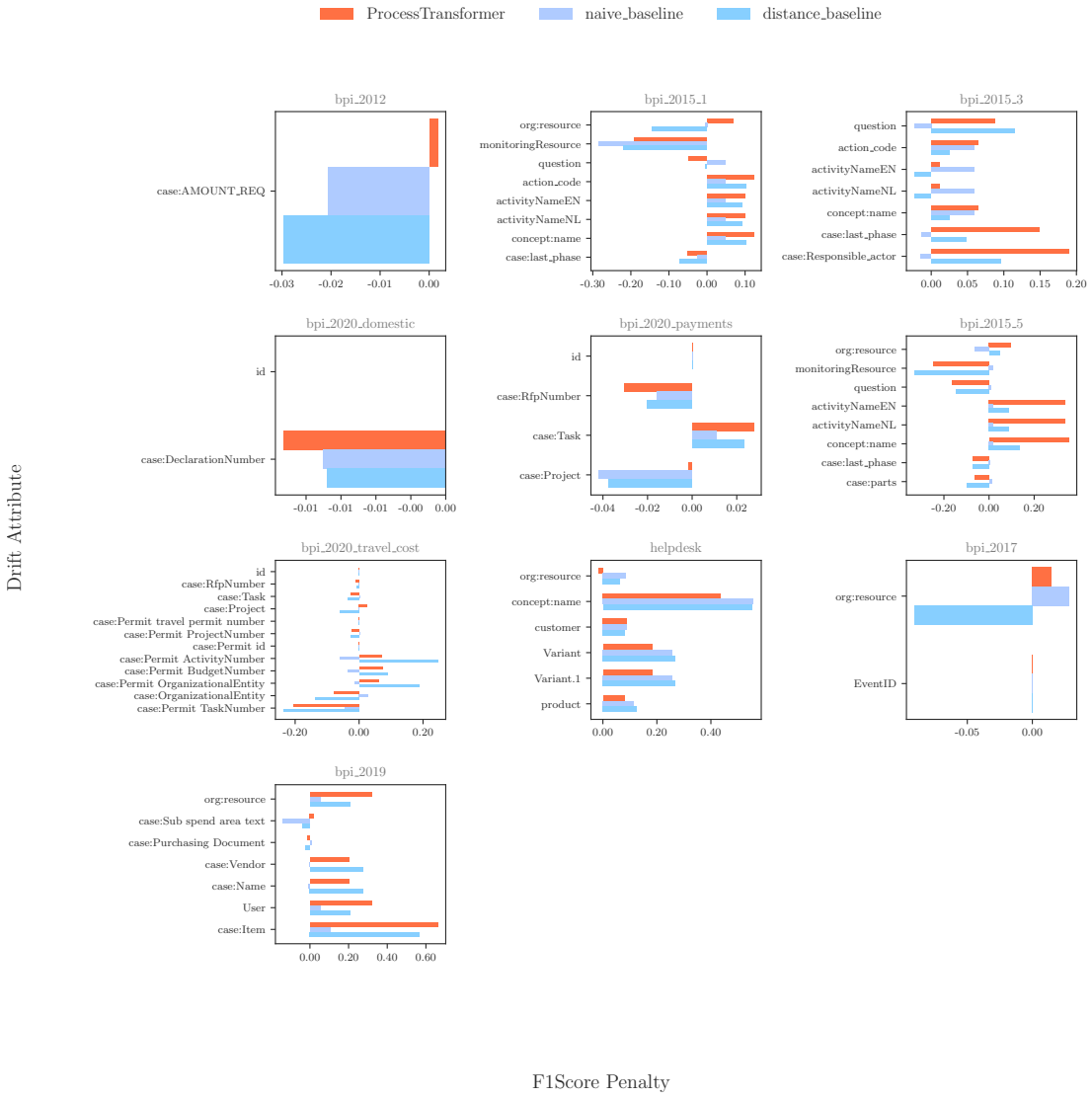


Figure 2: Comparison of the average reduction in mean absolute error score (MAE Penalty) across twelve event logs for test instances where some attribute value was not observed in the training data (attribute drift). Each subplot compares the average decrease in MAE (MAE penalty) for two approaches: ProcessTransformer (orange) and a baseline model (blue). The y-axis indicates the attribute, the x-axis shows the average MAE penalty. Higher MAE penalty values signify a model is more capable of dealing with attribute drift for the given attribute.

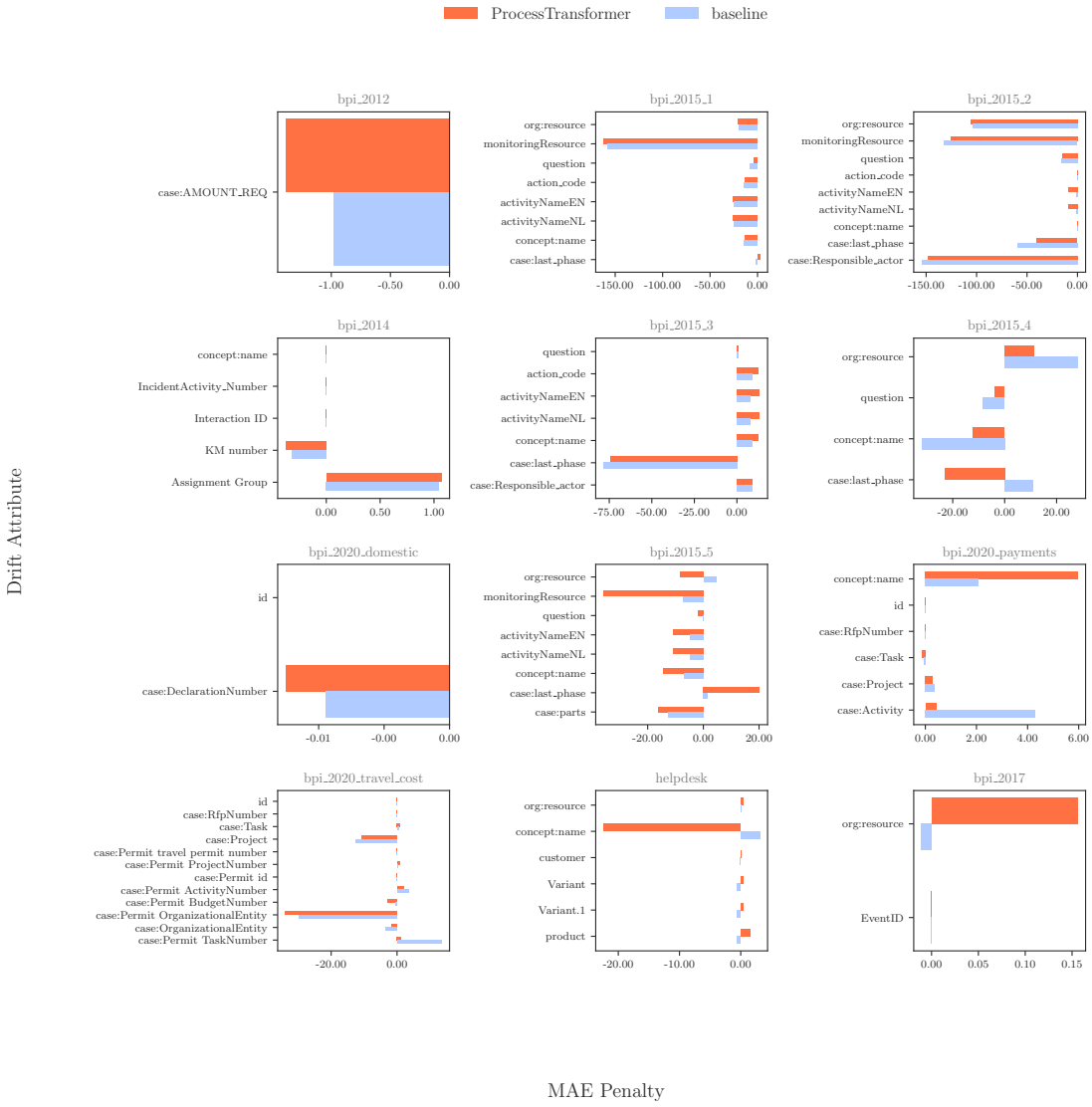


Figure 3: Comparison of predictive performance across eleven event logs grouped by the fraction complete of the test prefix. Each subplot compares F1-score for three approaches—ProcessTransformer (orange), a naive baseline (light blue), and a distance baseline (darker blue)-as the fraction completed increases (x-axis). Higher F1-scores signify better predictive performance.

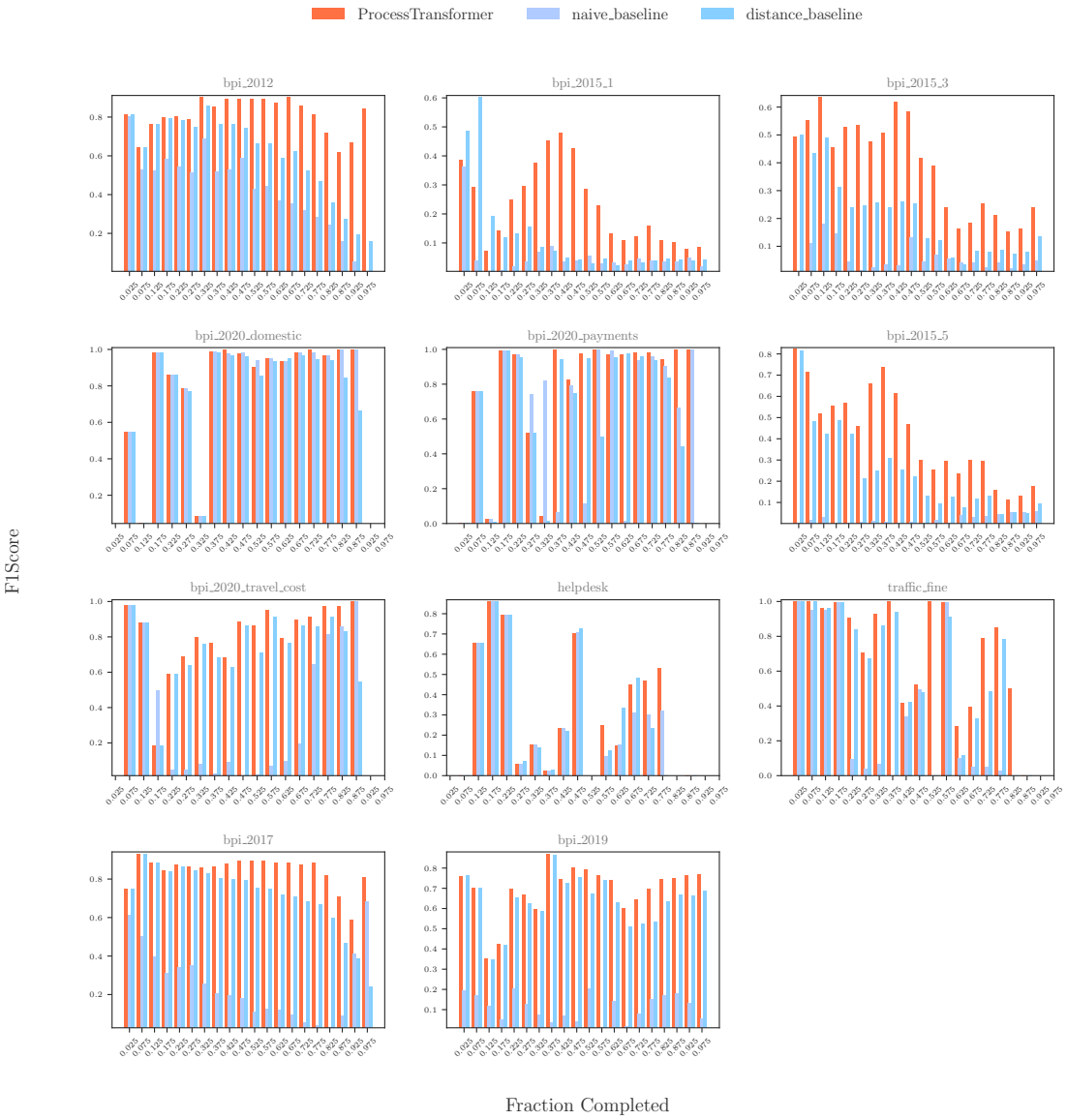
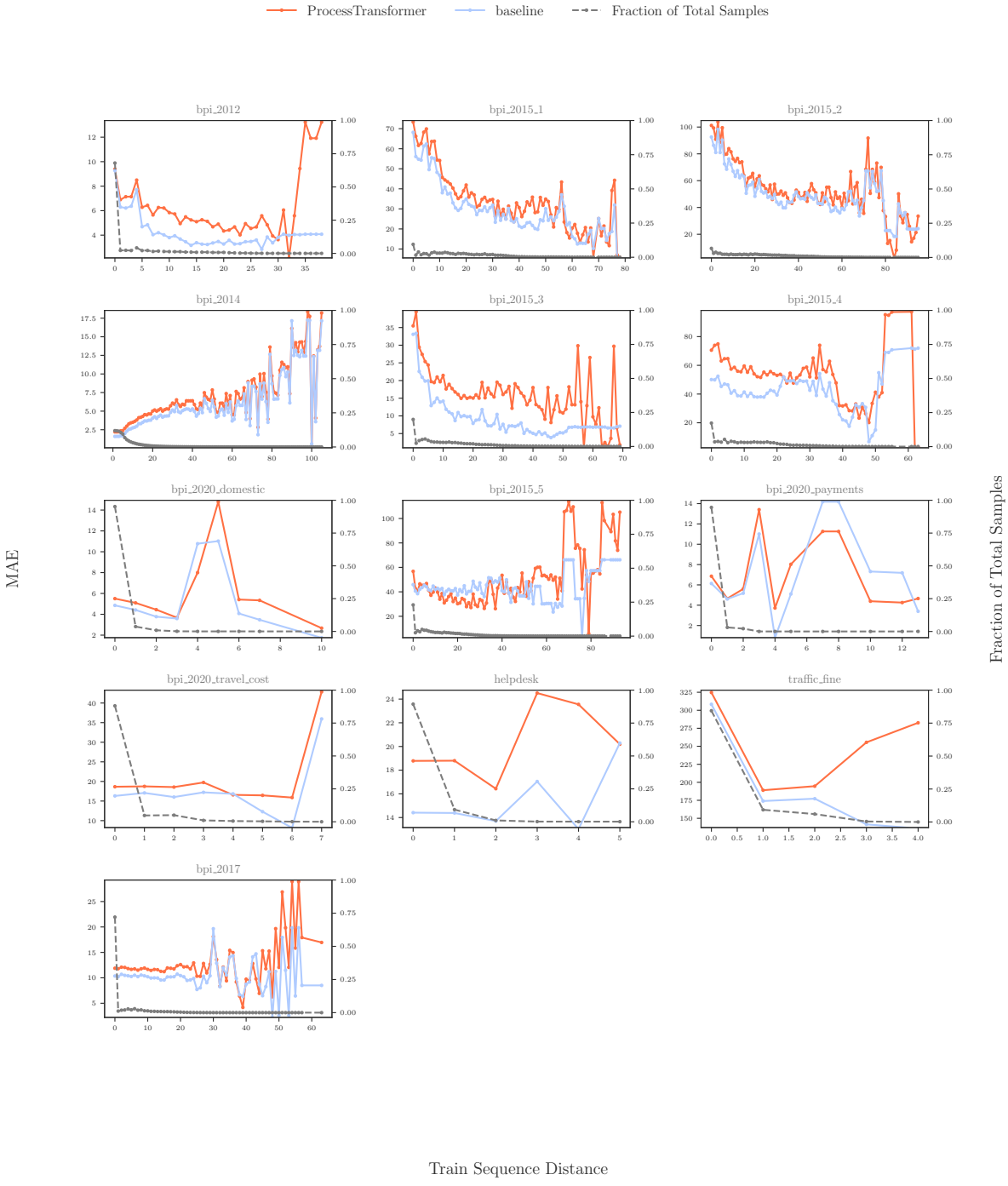


Figure 4: Comparison of predictive performance across thirteen event logs grouped by the fraction complete of the test prefix. Each subplot compares mean absolute error (MAE) for two approaches—ProcessTransformer (orange) and a baseline model (blue)—as the fraction completed increases (x-axis). Lower MAE values signify better predictive performance.

Figure 5: Comparison of predictive performance across eleven event logs grouped by the test prefix distance from the closest train prefix. Each subplot compares F1-score for three approaches—ProcessTransformer (orange), a naive baseline (light blue), and a distance baseline (darker blue)—as the training sequence distance increases (x-axis). The dashed gray line in each subplot indicates the fraction of total samples at each distance (right-hand y-axis). Higher F1-scores values signify better predictive performance.

Figure 6: Comparison of predictive performance across thirteen datasets grouped by the test prefix distance from the closest train prefix. Each subplot compares mean absolute error (MAE) for two approaches—ProcessTransformer (orange) and a baseline model (blue)—as the training sequence distance increases (x-axis). The dashed gray line in each subplot indicates the fraction of total samples at each distance (right-hand y-axis). Lower MAE values signify better predictive performance.



Appendix 3: Remaining time fraction completed results

Figure 7: Comparison of predictive performance across seven BPIC datasets grouped by the fraction complete of the test prefix. Each subplot compares mean absolute error (MAE) for three approaches—ProcessTransformer (orange), PGTNet (green), and a baseline model (blue)—as the fraction completed increases (x-axis). Lower MAE values signify better predictive performance.