Norwegian University of Life Sciences
School of Economics and Business

Philosophiae Doctor (PhD)
Thesis 2022:21

Essays on predictive and prescriptive process monitoring

Essays om prediktiv og preskriptiv prosessovervåking

Mike Riess

Essays on predictive and prescriptive process monitoring

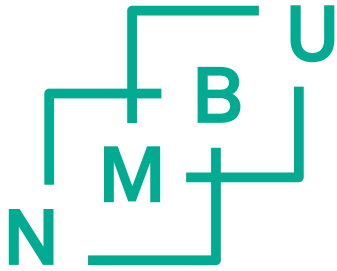
Essays om prediktiv og preskriptiv prosessovervåking

Philosophiae Doctor (PhD) Thesis

Mike Riess

School of Economics and Business
Norwegian University of Life Sciences

Ås (2023)



Thesis number 2022:21
ISSN 1894-6402
ISBN 978-82-575-1896-7

Supervisory team

Joachim Scholderer, Professor (main supervisor)
School of Business and Economics
Norwegian University of Life Sciences

Evaluation committee

Paul Grefen, Professor (1st opponent)
School of Industrial Engineering
Eindhoven University of Technology

Patrick Mikalef, Professor (2nd opponent)
Faculty of Information Technology and Electrical Engineering
Norweigan University of Science and Technology

Daumantas Bloznelis, Associate Professor (committee coordinator)
School of Business and Economics
Norwegian University of Life Sciences

Summary

This PhD thesis addresses problems related to proactive methods of decision support in business processes. These methods include predictive process monitoring, which aims to warn about potential problems before they occur, and prescriptive process monitoring which seek to proactively remedy predicted issues before they materialize. Four different studies are performed with the aim of improving methods within this area.

Paper one addresses the issue of early warning performance in predictive process monitoring. Specifically, this paper focuses on remaining cycle time prediction from open cases in business processes. The main goal of this paper is to understand how temporal weighting of the L1 loss function influence so-called earliness performance (the ability to make accurate early warnings). To investigate this, three different loss functions with temporal decay are introduced and evaluated across four real-world event-logs. This study also introduces a new aspect of performance evaluation of remaining time predictions, called Temporal Consistency (TC). The TC represents the degree to which a remaining time prediction model generates predictions that are monotonically decreasing as time passes. The results show that adding temporal decay to the L1 loss function can lead to better earliness performance. In particular, it was found that the proposed exponential temporal decay loss improved the earliness performance in two of the four evaluated settings. It was also found that all the evaluated loss functions had problems with respect to the temporal consistency performance criteria. This problem became most expressive for the longest traces with little support, where the remaining time prediction would change direction by a large amount.

Paper two offers an alternative to the traditional approach to model evaluation commonly used in predictive process monitoring, by proposing an open source simulation framework for the generation of synthetic event-log data. Firstly, a review of the current literature in this area was examined in order to provide an overview of current capabilities and potential gaps. From this review, a set of design criteria was formulated, and a new framework covering these areas was proposed. The resulting framework is based on well-known parametric distributions and intended for the generation of event-log data from theoretical business processes while providing

the capability to add systematic variation to the processes. The proposed framework is mainly intended for testing the influence of data-related hypotheses on the performance of models in predictive process monitoring. The framework enables systematic variation of process memory (in the context of a Markov chain), the entropy of workflows, activity duration distributions, and process stability. Detailed documentation on the implementation, as well as a demonstration of the framework, was performed. The resulting framework is open source and thereby freely available online.

Paper three addresses the issue of customer loyalty in customer service settings. In this study, a prescriptive method is proposed to improve customer loyalty by dynamically changing the priority of the queue in a customer service process. The proposed method uses predicted throughput/cycle time to further predict the conditional customer loyalty score after case closure (measured via Net promoter score). The proposed method is compared to the first-come first served (FCFS) queue discipline, as well as two predictive methods utilizing the shortest remaining time first (SRTF) and longest remaining time first (LRTF) disciplines. The methods are evaluated based on an agent-based simulation model, calibrated from historical data of a customer service process in a European internet and telecommunications services provider. The results show that the proposed method does improve simulated customer loyalty scores in situations with inadequate staffing. However, the proposed method yields similar results to that of the LRTF approach, as both methods rely on the prediction of case cycle time. Introducing a service level of a maximum of 60 hours of waiting time (to avoid starvation), caused all approaches based on predicted cycle time to have identical performance to that of FCFS.

Common for the methods studied in papers one and three is the need for adaptation (re-training) if the data-generating process changes over time. This is also referred to as concept drift, and can greatly reduce the performance of predictive and prescriptive methods if not addressed in time. Paper four thereby contributes to this area by performing a literature review on methods for drift adaptation using a family of optimization algorithms referred to as Metaheuristics. An overview of the found literature is provided through a qualitative analysis of frameworks in relation to selected theory within Automated machine learning, Data stream mining, and Concept drift. The results show that the most frequently used Metaheuristics are population-based methods such as Genetic Algorithms and Particle-Swarm Optimization, and that their utilization for drift adaptation varies from feature selection, hyper-parameter optimization to data window selection. General problems in terms of model and drift evaluation are found across the included literature, and suggestions for improvements in future research are made. Analyzing the temporal development across the found studies, it is found that the applications of Metaheuristics have developed from single Machine learning tasks such as feature selection to more advanced tasks such as full model selection.

Sammendrag

Denne doktorgradsavhandlingen tar for seg problemer knyttet til proaktive metoder for beslutningsstøtte i forretningsprosesser. Disse metodene inkluderer prediktiv prosessovervåking, som tar sikte på å advare om potensielle problemer før de oppstår, og preskriptiv prosessovervåking som forsøker å proaktivt rette på predikerte problemer før de materialiserer seg. Det er utført i fire ulike studier med mål om å forbedre metoder innen dette område.

Artikkel en tar opp spørsmålet om ytelse for tidlig varsling i prediktiv prosessovervåking. Spesifikt fokuserer denne artikkelen på prediksjon av gjenværende syklustid fra åpne saker i forretningsprosesser. Hovedmålet med denne oppgaven er å forstå hvordan tidsmessig vekting av L1-tapfunksjonen påvirker såkalt tidlighetsytelse (evnen til å gi nøyaktige tidlige advarsler). For å undersøke dette, introduseres tre forskjellige tapsfunksjoner med tidsforfall som evalueres på tvers av fire hendelseslogger fra faktiske bedrifter. Studien introduserer også et nytt aspekt på ytelsesevaluering av gjenværende tidsprediksjoner, kalt temporal konsistens (TC). TC representerer i hvilken grad en prediksjonsmodell for gjenværende tid genererer prediksjoner som avtar monotont etter hvert som tiden går. Resultatene viser at å legge til tidsmessig forfall til L1-tapfunksjonen kan føre til bedre tidlighetsytelse. Spesifikt ble det funnet at eksponentiell tidsforfall forbedret tidlighetsytelsen i to av de fire evaluerte innstillingene. Det ble også funnet at alle tapsfunksjoner som ble evaluert hadde problemer med hensyn til den TC. Mer spesifikt ble det funnet at for lengre sekvenser med lite representativitet i data blir dette problemet mest uttrykksfullt. Rent praktisk betyr dette at prediksjonene i disse tilfellene endrer retning i en høy grad.

Artikkel to tilbyr et alternativ til den tradisjonelle tilnærmingen av modellevaluering som vanligvis brukes i prediktiv prosessovervåking. Dette opnås ved å foreslå et simuleringsrammeverk med åpen kildekode for generering av syntetiske hendelsesloggdata. Først undersøkes en gjennomgang av gjeldende litteratur på dette området. Dette er for å gi en oversikt over egenskaber og potensielle hull. Fra denne gjennomgangen blir et sett med designkriterier først formulert, og et nytt rammeverk som dekker disse områdene foreslås da. Rammeverket er basert på velkjente parametriske distribusjoner og beregnet for generering av teoretiske forretningspros-

esser samtidig som det muliggjør muligheten til å legge til systematisk variasjon. Det foreslåtte rammeverket er ment for testing av datarelaterte hypoteser om ytelsen til modeller innen prediktiv prosessovervåking. Rammeverket muliggjør systematisk variasjon av prosessminne (i sammenheng med en Markov-kjede), entropi av arbeidsflyter, aktivitetsvarighetsfordelinger og prosessstabilitet. Det utføres en detaljert dokumentasjon på gjennomføringen, samt en demonstrasjon av rammeverket. Det resulterende rammeverket er åpen kildekode og dermed fritt tilgjengelig online.

Artikkel tre tar opp spørsmålet om kundelojalitet i kundeservicesettinger. I denne studien foreslås en preskriptiv metode for å forbedre kundelojalitet ved dynamisk å endre køens prioritet i en kundeserviceprosess. Den foreslåtte metoden bruker spådd gjennomstrømning/syklustid for ytterligere å forutsi den betingede kundelojalitetsscore etter saksavslutning (målt via Net promoter-score). Den foreslåtte metoden sammenlignes med først-til-mølla-disiplinen (FCFS), samt to prediktive metoder som bruker den korteste gjenværende tid først (SRTF) og lengste gjenværende tid først (LRTF) disipliner. Metodene er evaluert basert på en agentbasert simuleringsmodell, kalibrert fra historiske data fra en kundeserviceprosess i en europeisk internett- og telekommunikasjonsleverandør. Resultatene viser at den foreslåtte metoden forbedrer simulerte kundelojalitetsscore i situasjoner med utilstrekkelig bemanning av kundeserviceprosessen. Imidlertid gir den foreslåtte metoden lignende resultater som LRTF-tilnærmingen, ettersom begge metodene er avhengige av prediksjonen av sakssyklustid. Innføring av et servicenivå på maksimalt 60 timers ventetid (for å unngå at kunder blir værende bak i køen) reduserte ytelsen til alle tilnærminger basert på predikert syklustid til å være identisk med førstemann til mølla.

Felles for metodene som er studert i artikkel en og tre er behovet for tilpasning (gjenopplæring) dersom den datagenererende prosessen endres over tid. Dette blir også referert til som konseptdrift, og kan i stor grad redusere ytelsen til prediktive og foreskrivende metoder hvis de ikke blir adressert i tide. Artikkel fire bidrar dermed til dette området ved å utføre en litteraturgjennomgang om metoder for tilpasnins av konseptdrift ved bruk av en familie av optimaliseringsalgoritmer referert til som metaheuristikker. En oversikt over funnet litteratur gis gjennom en kvalitativ analyse av rammeverk i forhold til utvalgt teori innen automatisert maskinlæring, datastrømmining og konseptdrift. Resultatene viser at de mest brukte metaheuristikkene er populasjonsbaserte metoder som genetiske algoritmer og partikkelsvermoptimalisering, og at deres utnyttelse for drifttilpasning varierer fra valg av variable, hyperparameteroptimalisering til datavinduvalg. Generelle problemer når det gjelder modell- og driftevaluering finnes på tvers av den inkluderte litteraturen, og forslag til forbedring i fremtidig forskning blir derved gitt. Ved å analysere den tidsmessige utviklingen på tvers av de funnet studiene, er det funnet at bruken av metaheuristikker har utviklet seg fra enkle maskinlæringsoppgaver som valg av variable til mer avanserte oppgaver som full modellvalg.

Acknowledgements

First of all, I would like to thank my supervisor Joachim Scholderer who have been a big help during the whole process of this PhD project. I have very much enjoyed our conversations and collaboration during this period. I am deeply thankful for the many opportunities you have given me to learn and grow as a researcher. I also appreciate all the support and advice you have given me in hard periods.

Next, I would like to thank the School of Economics and Business at NMBU for the opportunities you have given me to help teach and supervise students throughout the PhD-period. This has been both exciting and meaningful. Thanks to Pål Bjørnhaug Johannsen and William Irving for their support, and to Erik Henning Edvardsen and Andreas Rosendahl Hansen for their help with proof-reading of this dissertation.

A big thank you to my parents Gitte, Carsten and Anders: You have taught me important values, which have brought me where I am today, and you have supported me every time I needed it. Also a thank you to Wini who have been full of good advice and support when it was needed the most.

Finally, a deep-felt thank you to the woman in my life, Monika, who gives me energy, advice, and who made the two years of pandemic and home-office bearable. I would like to dedicate this work to you, as you have been so understanding and supporting during the whole PhD-period.

Oslo, Mar,
2023

Mike Riess

Table of Contents

Summary . . . iii
Sammendrag . . . v
Acknowledgements . . . ix
Table of Contents . . . xi
List of Figures . . . xiii
List of Tables . . . xv
List of Acronyms . . . xvii

List of Publications **xvii**

1 Introduction **1**
1.1 Problem statement . . . 2
1.2 Theory and previous research . . . 4
1.2.1 Business processes . . . 4
1.2.2 Business process management . . . 5
1.2.3 Descriptive, predictive and prescriptive analytics . . . 9
1.2.4 Predictive process monitoring . . . 11
1.2.5 Prescriptive process monitoring . . . 14
1.2.6 Business process simulation models . . . 17
1.2.7 Concept drift . . . 18

2 Research questions **21**

3 Methodology **27**
3.1 Methodology in Machine learning research . . . 27
3.2 Overall research framework . . . 28
3.2.1 Data . . . 28
3.2.2 Methods . . . 30
3.3 Individual discussion of used methodology . . . 31
3.3.1 Paper 1 . . . 32
3.3.2 Paper 2 . . . 32
3.3.3 Paper 3 . . . 33
3.3.4 Paper 4 . . . 34

4 Main contributions **37**
4.1 Paper 1 . . . 38
4.2 Paper 2 . . . 39
4.3 Paper 3 . . . 40
4.4 Paper 4 . . . 41

5 Reflection and discussion **43**

References **45**

Appended Papers **53**
Paper I . . . 55
Paper II . . . 73
Paper III . . . 99
Paper IV . . . 135

List of Figures

1.1 Example of process discovery using the Disco process mining software (Fluxicon BV, 2022). 10
1.2 Analytics-types in relation to time and business value. Source: Adapted from (Lepenioti et al., 2020) and (Krumeich et al., 2016). 11
1.3 Predictive process monitoring example. 12
1.4 Prescriptive process monitoring example illustration. 15

3.1 Methodological framework. 29

List of Tables

1.1 Overview of phases in the BPM life cycle. 6
1.2 The devil's quadrangle examples (Jansen-Vullers et al., 2007; Dumas et al., 2018). 8
1.3 Example event-log in a fictive customer service unit. 9

List of Publications

This thesis is based upon the following appended papers, which will be referred to as papers 1-4 throughout the text.

Paper I

Riess, M. (2023c). Remaining cycle time prediction: Temporal loss functions and prediction consistency. Manuscript submitted to *Nordic Machine Intelligence*.

Paper II

Riess, M. (2023a). A parametric simulation framework for the generation of event-log data. Manuscript submitted to *Simulation*.

Paper III

Riess, M. and Scholderer, J. (2023). Customer-service queuing based on predicted loyalty outcomes. Manuscript submitted to *Decision Support Systems*.

Paper IV

Riess, M. (2023b). Automating model management: A survey on metaheuristics for concept-drift adaptation. Revised version of paper published in Journal of Data, Information and Management (2022), Vol. 4, 211–229.

1. Introduction

Modern business process management relies on a fast flow of information between managers, specialized workers and customers to ensure that the best possible output is generated at all times. Traditionally, retrospective tools such as statistical process control (SPC) (Levinson, 2010), Lean and Six Sigma (Pepper and Spedding, 2010) have been used to achieve process excellence.

However, as business processes have over the past 30+ years been increasingly digitized due to rapid growth in the capabilities of information technology, new opportunities have appeared (Diao et al., 2016). More specifically, as process-aware information systems (PAIS) (van der Aalst, 2016) have been adopted across all industries, the combination of historical data, dashboards and machine learning algorithms (Hastie et al., 2001) now offer the possibility to *proactively* react to problems faster and smarter than ever, commonly referred to as *analytics* (Davenport et al., 2006). Being *reactive* in nature, the traditional tools of workforce management and process standardization no longer provide a competitive advantage in itself (Diao et al., 2016).

This Ph.D. thesis study *proactive* methods of decision support in business process management, also known as *predictive and prescriptive process monitoring*. Common for these methods is that they rely on Machine learning algorithms to predict future states of a business process, in order to alert or recommend actions for process managers. With the aim of improving methods within this area, this project focuses on four key issues: 1) Alerting managers as early as possible, 2) Understanding strengths and weaknesses of predictive monitoring systems, 3) Improving customer loyalty via model-based queue management in customer service, 4) Gaining an overview of methods for maintaining models used in predictive monitoring systems.

This thesis is structured as follows: In the remainder of chapter 1, the problem statement and research objectives are firstly presented. Next, relevant theory and previous research in the context of the included research are presented. In chapter 2, the research questions guiding the work of the four individual studies of this thesis are motivated and presented. Chapter 3 provides an overview of the methodological

framework, in addition to individual discussions of the methodology used. Chapter 4 presents the main contributions of the four studies, and in chapter 5, a reflection and discussion of the project contributions is made.

1.1 Problem statement

In predictive process monitoring (Di Francescomarino and Ghidini, 2022), multiple approaches to the prediction of remaining cycle/throughput time (time it takes to complete a case such as a customer service issue) exist (Verenich et al., 2019). Common for the best-performing Neural Network-based methods is that they are optimized using the L1 (Mean Absolute Error) loss function (Rama-Maneiro et al., 2020). A performance aspect that is often important for organizations trying to create value from predictive process monitoring is referred to as *earliness*, which takes the timing of the errors of predicted cycle time into account: a model with good earliness has relatively lower errors at early time steps, compared to other models (Verenich et al., 2019). Another aspect that might be of importance is the consistency of the predictions: i.e., that they represent the natural development of time (remaining cycle-time is monotonically decreasing as time passes). However, currently, cycle time prediction models have not yet been evaluated from this aspect. Furthermore, as the best performing models in this area all use the time-invariant L1 loss, it is currently not understood how temporal weighting of the loss function would influence performance from the earliness perspective.

As the models presented in the field of predictive process monitoring are mostly evaluated using publicly available data (Verenich et al., 2019; Rama-Maneiro et al., 2020), understanding the relationships between model performance and particular data characteristics has been limited. As business processes of the same type tend to vary across organizations, evaluating the model using one example of a given process is not nearly enough to understand how this type of process generally influences model performance. In this case, one might need to sample data from multiple processes (distributions) to control for variation between distributions. However, this is a demanding task as: 1) There are a limited amount of publicly available data sets available 2) One is not guaranteed to find enough instances of the particular process type 3) One has no control over the variation within the distribution of each of the publicly available data sets. For these reasons, the research in predictive process monitoring is mainly based on benchmarking performance across a set of commonly used datasets (Mannhardt et al., 2015; Mannhardt and Blinde, 2017; La Rosa and Soffer, 2013; Teniente and Weidlich, 2018). For comparing relative model performance, this approach is unproblematic, but if the aim is to achieve a general understanding of model robustness in relation to characteristics of the data-generating process, this is inadequate.

In the related field of prescriptive process monitoring, multiple studies exist which aim to improve process performance such as reducing the average cycle time, costs or

contractual violations. These approaches are mainly evaluated using historical data, and most often based on publicly available data used in academic competitions, where some contextual information could be missing, unless disclosed together with the data. Queue dynamics and particularly the topic of queue priority in a customer service context, have not yet been addressed in the prescriptive process monitoring literature (Kubrak et al., 2022). However, in computer science and operations research, queue scheduling algorithms and their effect on process dynamics are considered well-understood (Omar et al., 2021). Yet, to the best knowledge of the author of this thesis, no work currently exists on predictive queue scheduling, with the goal of improving quality performance measures such as customer loyalty. From a customer relationship management perspective, loyal customers are important as they have the highest lifetime value, and thereby represent the highest future profits for a company (Blattberg et al., 2008). A further understanding of how customer loyalty can be affected via prescriptive process monitoring, would therefore be a contribution in this area.

Common for these model-based approaches in predictive and prescriptive process monitoring is that when implemented in an organization, they might be subject to so-called concept drift (Tsymbal, 2004). This means that the distribution they are trained from could have changed over time, effectively decreasing the performance of the model and thereby its value to the organization. In these cases, the model needs to be re-trained from new data (the new distribution). In these situations, there is a trade-off between operating costs and performance, as one might simply re-train a model every day, however, this approach has the disadvantage of significantly higher operating costs (Gama et al., 2013). Drift detection and automated adaptation can therefore be of benefit in scenarios where the cost of re-training is high. One such scenario is the remaining time prediction approaches based on deep learning. However, in the field of predictive process monitoring, only a few studies have currently been made in terms of automated drift adaptation (Baier et al., 2020; Maisenbacher and Weidlich, 2017). An overview of current approaches and their methodology might therefore be helpful for future research on automated drift adaptation in predictive process monitoring.

The overall objective of this Ph.D. project is to:

- Improve methods for predictive and prescriptive process monitoring.

This thesis, therefore, aims to accomplish the following set of research objectives:

1. Understand how temporal weighting of loss functions influences the earliness performance of remaining time prediction models.
2. Propose a simulation framework that improves the understanding of model performance by enabling researchers to specify and generate synthetic process data for model evaluation.
3. Understand how customer loyalty can be influenced via predictive queue pri-

oritization in a customer service process.

4. Understand how previous literature has studied automated adaptation of machine learning models in settings with concept drift.

1.2 Theory and previous research

In this chapter, research areas that forms the foundation of predictive and prescriptive process monitoring will firstly be introduced through a description and discussion of aspects relevant to this thesis. This will then be followed by a discussion of previous research within each of the four problem areas outlined in the problem statement: Predictive process monitoring, Prescriptive process monitoring, Business process simulation models and Concept drift.

1.2.1 Business processes

Davenport and Short (1990) formally define a business process as *a set of logically-related tasks performed to achieve a defined business outcome*. Business processes might take many forms, but what is common is that they have *customers*, and might cross organizational boundaries (units within the organization) (Davenport, Short, et al., 1990). Furthermore, a business process generates value by taking an *input* and transforming it into a given *output* that is of value to internal or external customers (Porter, 1985; Slack et al., 2016). Business processes vary across and within organizations, however, some attempts have been made to formalize the structure of the most common types of processes. An example is the end-to-end process definitions by the American Productivity and Quality Center (APQC) (Center), 2022), which currently contain 31 generic processes with activities common within service and production sectors.

A particular type of business processes relevant to this thesis, is *service processes*, which deliver value to internal or external customers, not by creating goods or products, but by performing a set of actions that create value for the customer. In modern enterprises, every service process is relying on information technology to be operational, and hence the rise of the field of Information Technology Service Management (ITSM) (Galup et al., 2009). Similar to the generic end-to-end processes described by APQC, the service sector has dedicated (often commercial) frameworks and process maps for service management, such as Information Technology Information Library (ITIL) (Agutter, 2020), and Business Process Framework (eTOM) (TMForum, 2023).

A service process might take a customer issue in the form of an email and transform it into a solution, hopefully resulting in a satisfied customer. Based on the end-to-end process definitions by the American Productivity and Quality Center (Center), 2022), this type of process is known as the *issue-to-resolution* process.

Particularly in ITSM, it is common to have *Service Level Agreements (SLA)* (Slack et al., 2016), which specify measurable performance such as waiting time for the internal or external customer before the issue is resolved. A key aspect of service processes is thereby the aspect of time, which is influenced by the combination of the capacity of the process and the inflow of customers.

This can formally be represented as a stochastic *queue system*, which using *Kendall notation* has the following form (Slack et al., 2016): $M/M/s/k$, where the first M denote the distribution of arrivals, the second M the distribution of processing times, s the number of servers (agents) in the process, and k the maximum number of customers waiting. Furthermore, a queue system of a service process might have a queue *discipline*, which denotes the order in which issues (or cases) are prioritized (Omar et al., 2021); First-Come First Served (FCFS), Last-Come First Served (LCFS), Shortest Job First (SJF), Longest Job First (LJF) and Service In Random Order (SIRO). The queue discipline might be either preemptive or non-preemptive, where preemptive queuing allows a given issue to be temporarily abandoned in favor of another, in order to shorten the queue length or due to higher priority of another queue (Omar et al., 2021). Mathematical representations of queue systems combined with Monte-Carlo simulation (Rubinstein and Kroese, 2016) can be used as tools in the management of service operations, answering questions such as: How many agents are needed to fulfill a given service level to the customers, or which impact changes to the queue discipline might have on the average waiting time (Slack et al., 2016).

1.2.2 Business process management

The origins of business process management lie in a stream of literature known as *Business process re-engineering (BPR)* (Davenport, Short, et al., 1990), which focuses on process improvement through the re-design of existing processes. Originating in the early 1990s, a key enabler in these efforts was the digitization of processes with the introduction of information technology. The review by (O'Neill and Sohal, 1999) found that literature in BPR mainly focused on *tools* such as; Process visualization, process mapping, change management, benchmarking, and customer focus to improve processes.

According to (Dumas et al., 2018), the field of BPR became less popular in the late 1990s due to: 1) Concept misuse, as organizations would name every change program (including down-sizing) as BPR, 2) Early literature in the field encouraged radical rather than small changes in every BPR project, which was not appropriate in all cases, 3) Immaturity of support systems, as process logic was often hard-coded in customized IT systems and could thereby not easily be changed. The emergence of empirical studies showing performance gains by organizations adopting the process-centered management, compared to those who did not (McCormack, 1999), combined with the introduction of supporting Information Systems (IS) mo-

Phase	Source	Description
Identification	(Dumas et al., 2018)	Identification of relevant processes related to a business problem in focus. The goal is to select a process to focus on in the remainder of the life cycle.
Process discovery	(Dumas et al., 2018)	Modelling and documentation of the as-is process.
Process analysis or diagnosis	(Dumas et al., 2018; van der Aalst, 2016)	Issues related to the as-is process is measured and analyzed.
Process redesign	(Dumas et al., 2018; van der Aalst, 2016)	Identification of changes to the process that can help improve the performance issues discovered in the previous step.
Process implementation	(Dumas et al., 2018; van der Aalst, 2016)	Transformation of the as-is process into the to-be process, based in the identified changes in the previous step.
Process monitoring	(Dumas et al., 2018; van der Aalst, 2016)	Collection and analysis of data to determine process performance.
Process adjustment	(van der Aalst, 2016)	Predefined controls are used to adapt or re-configure the running process, without a need for redesign.

Table 1.1: Overview of phases in the BPM life cycle.

tivated further BPR-related research, which formed the basis of Business process management (Dumas et al., 2018).

The field of Business process management (BPM) focuses on process design and improvement as part of a continuous life cycle of the process in question (Dumas et al., 2018). The BPM life cycle consists of a set *phases*, however, the number of (and which) phases differ in the literature. Where (Dumas et al., 2018) describe 6 phases, (van der Aalst, 2016) only present 5. Furthermore, there is only a partial overlap between the two frameworks. Consequently, an overview of the phases in both frameworks and their overlap is presented in table 1.1. In each of the two frameworks, the individual order is the same as the one in which they are listed in table 1.1. Common for both frameworks, is that it is a *cycle*, in that the process repeats itself once the last phase is finished.

As illustrated by the BPM life cycle in Table 1.1, business process management span all phases in the life cycle of a business process. The field of business process management have thereby become a umbrella for multiple other streams of research.

As this thesis is focused on *predictive and prescriptive process monitoring*, process monitoring, process-aware information systems and process mining will be discussed in the following sections.

Process monitoring

Value creation in a business process can be measured and monitored using metrics also referred to as *key performance indicators* (KPI) (Fitz-Gibbon, 1990; Dumas et al., 2018). van der Aalst, 2016 relate these to three dimensions, namely: *time*, *costs* and *quality*, where Dumas et al., 2018 propose process *flexibility* as a fourth dimension. As discussed by Dumas et al., 2018, these four dimensions are closely related, such that changing either of these will have an effect on one or more of the remaining dimensions (in an unfavorable manner). This is also referred to as *the devil's quadrangle* in the area of business process re-engineering (Jansen-Vullers et al., 2007).

A list of examples from each of the four dimensions provided in (Jansen-Vullers et al., 2007; Dumas et al., 2018), can be seen from Table 1.2. The authors of (Dumas et al., 2018) divide process monitoring into two categories: *online* and *offline*. Online process monitoring is in this context referred to as the monitoring of cases while they are active, whereas offline monitoring refers to the analysis of historical case data.

Process-aware information systems

What defines a Process-aware Information System (PaIS), is the notion that it is aware of the process in which it is used, and not specific to any single activity (van der Aalst, 2016). Examples of process-aware information systems are Customer Relationship Management (CRM) systems, which facilitate planning and management of interactions with customers, as well as Enterprise Resource Planning (ERP) systems which facilitate more general processes in the value chain. A key property of PaIS are that they produce data that can be used in the analysis of processes. Such data is commonly referred to as event-log data, which most often includes a case identifier, an activity, a time stamp, and a resource (van der Aalst, 2016). The content of these event-logs might vary across systems and process types, as well as the meaning of a *case* and an *activity*. An example of an event-log can be seen from table 1.3.

In this fictive example from a *issue-to-resolution* process of customer service unit, the meaning of the columns are as follows (going from left to right): The unique identifier of the case, the general topic of the issue, the activity that was performed, the time when the activity was performed, and finally the name of the resource (agent) that performed the activity.

Dimension	Examples
Time	• Waiting time: – *Time between arrival and processing of a case* – *The idle time between processing of two tasks* • Cycle time/throughput time: – *The time from start to completion of a case*
Costs	• Operating costs: – *Labor costs* – *Training costs* • Inventory costs: – *Costs of keeping goods and materials needed*
Quality	• Performance: – *Customer satisfaction* – *Customer loyalty* • Conformance: – *Product quality* – *Compliance with internal rules*
Flexibility	• Labor flexibility: – *Ability of a worker to perform multiple task types* • Mix flexibility: – *Ability to perform multiple case types* • Volume flexibility: – *Ability to adapt to changes in volume of cases*

Table 1.2: The devil's quadrangle examples (Jansen-Vullers et al., 2007; Dumas et al., 2018).

Process mining

The field of process mining can be seen as a subset of business process management (Dumas et al., 2018), and has evolved from the introduction of process-aware information systems across industries. Process mining is defined as a set of tools that can aid in the BPM life cycle discussed earlier (van der Aalst, 2016). A central element in process mining is the notion of *process models*, which are abstractions of the real behavior in business processes. Process mining uses algorithms for so-called *process discovery*, which involves the automatic generation of an abstract process model based on event-log data. These constructed models can then be used for performance and conformance analysis (van der Aalst, 2016). The model-based performance analysis enables the identification of bottlenecks between activities, and can provide a visual understanding of the overall flow in the process. Confor-

Case ID	Case topic	Activity	Timestamp	Resource
1001	Invoice	Email interaction	01-01-2019 15:01	Lars K.
1001	Invoice	Phone interaction	02-01-2019 16:04	Lars K.
1001	Invoice	Send new invoice	04-01-2019 16:58	Lars K.
1002	Service upgrade	Email interaction	01-01-2019 12:01	Bjarne G.
1002	Service upgrade	Phone interaction	03-01-2019 13:10	Bjarne G.
1002	Service upgrade	Change data	03-01-2019 14:15	Bjarne G.
1002	Service upgrade	Email interaction	04-01-2019 09:35	Jeppe V.

Table 1.3: Example event-log in a fictive customer service unit.

mance analysis refer to the degree to which the observed behavior in the process reflects the *expected* or *desired* behavior. An outcome of these two types of analysis might be requirements for process adjustment or re-design as previously described in table 1.1.

An example of a discovered process model from the publicly available Helpdesk data (Verenich, 2016) can be seen from Figure 1.1. The orange boxes represent activities, and the arrows the transitions between activities observed in the event-log data. In this example, the waiting time between activities cannot be assessed as only the time stamp of the beginning of the activities is present in the event-log data. The time information between activities thereby represent both activity duration and waiting time between activities. If time stamps from both the start and end of the activities were present in the data, arrows would represent the average waiting time between activities, and the average duration would be represented within each box in the process model in Figure 1.1. As seen by this example, this type of process analysis is also limited by the amount of information captured by the PaIS.

A key goal of process mining is to provide operational support via process analysis and monitoring, which have led to two new streams of literature, namely; *predictive* and *prescriptive* process monitoring which will be further introduced in sections 1.2.4 and 1.2.5.

1.2.3 Descriptive, predictive and prescriptive analytics

In the classic 2006 Harvard Business Review article: *Competing on analytics* (Davenport et al., 2006), the discipline of analytics was described through a set of use-cases observed in American companies at the time. What is common for companies *competing on analytics* is the utilization of data, information systems and statistical modelling to drive decision making at all levels in businesses (Davenport et al., 2006). Since this paper, progress in Machine learning and the widespread adoption of information systems (such as PaIS discussed earlier) have enabled a significant body of literature within model-based managerial decision support (published in journals such as MIS Quarterly: Management Information Systems, Decision Support Systems, International Journal of Information Management, etc.), which can

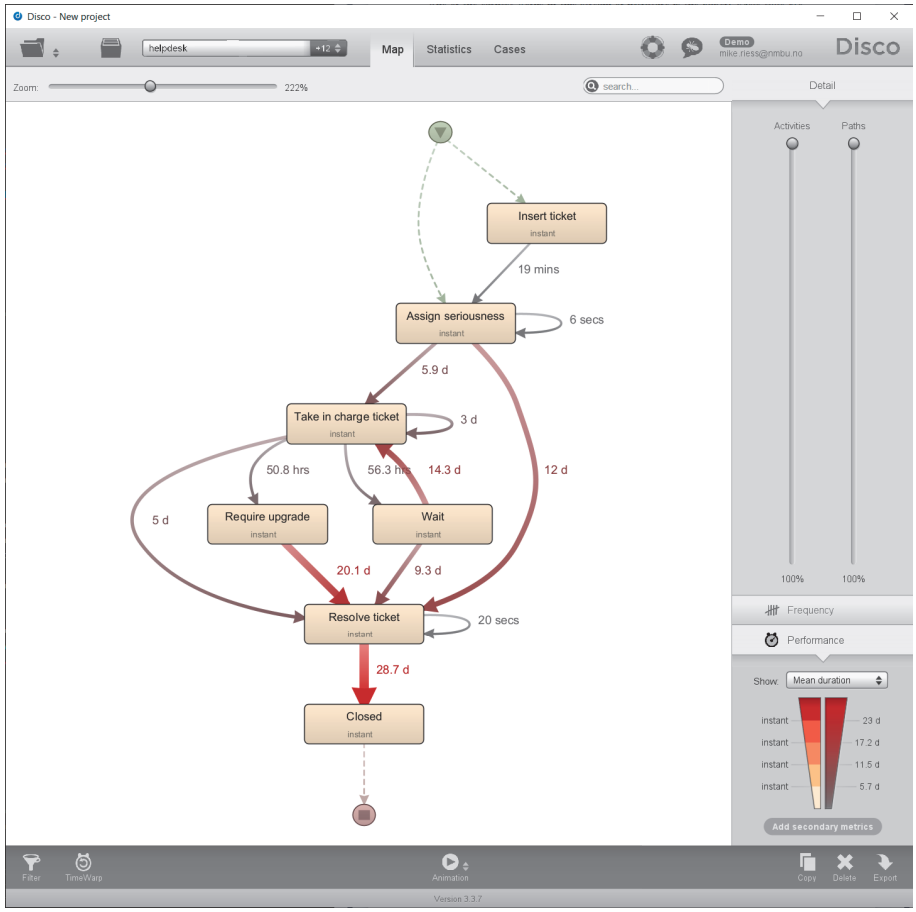


Figure 1.1: Example of process discovery using the Disco process mining software (Fluxicon BV, 2022).

all be classified as *analytics*.

From a high level, the authors in (Lepenioti et al., 2020) discuss that analytics can generally be divided into three different types: Descriptive, predictive and prescriptive. The first type, *descriptive* analytics, is reactive in nature as it focuses on past events, and according to (Lepenioti et al., 2020), the goal is most often to answer the questions: *What has happened?* or *what is happening?* (Lepenioti et al., 2020). Descriptive analytics thereby support managerial decision making by either looking backwards in time, or focusing on what is happening at the moment. *Predictive* analytics focuses on predicting future events, and thereby proactive while aiming to answer typical questions such as; *What will happen?*. Finally, *prescriptive* analytics aim to answer questions such as: *What should I do?* (Krumeich et al., 2016). An overview of the three types of analytics can be seen in Figure 1.2, which is an adapted version of the models presented in (Lepenioti et al., 2020; Krumeich

et al., 2016).

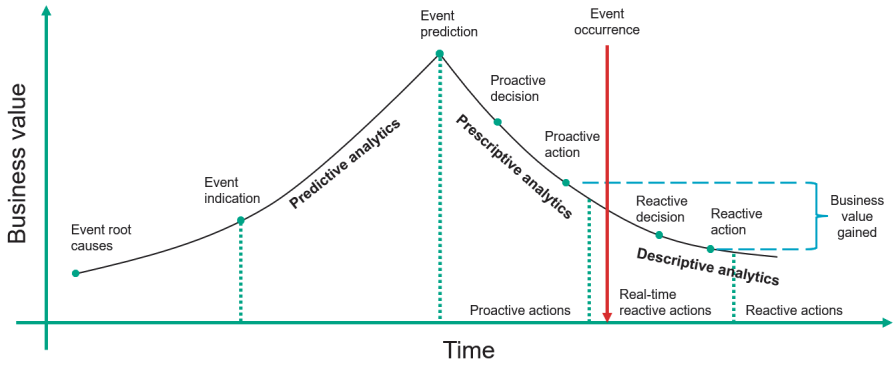


Figure 1.2: Analytics-types in relation to time and business value. Source: Adapted from (Lepenioti et al., 2020) and (Krumeich et al., 2016).

As illustrated in this Figure, potential business value gained lies between a *proactive* and *reactive* action (marked with blue on the right). Going back to the *issue-to-resolution* example earlier, while *descriptive analytics* might give an in-depth understanding of historical or current process performance, *predictive analytics* might give early warnings future performance. This will then enable a proactive decision, based on alternatives provided from *prescriptive analytics*. In other words, the *predictive* component enables the *prescriptive*, which could then lead to full automation and self-correction of the process. In the following, relevant literature within predictive and prescriptive process monitoring will be discussed.

1.2.4 Predictive process monitoring

The literature on predictive analytics in relation to business processes is generally referred to as *predictive process monitoring*. A predictive process monitoring system is based on a machine learning model (Hastie et al., 2001), which is *trained* from event-log data residing in a PaIS. Figure 1.3 illustrates the steps involved in generating a predictive process monitoring model (top), as well as how one such might be deployed in a running business process (bottom). The development process is similar to other machine learning models (see (Chapman et al., 2000)), with the exception of the data pre-processing. As event-log data has multiple rows per unit of observation (events per case), encoding of the data is important for predictive process monitoring (see (Verenich et al., 2019) for a detailed overview).

One of the earliest works in this stream of literature is the paper; *Cycle time prediction: When will this case finally be finished?* (van Dongen et al., 2008). In this paper the authors presented a non-parametric approach to predicting the remaining cycle-time from partially observed cases. The authors used event-log data from an administrative process within a Dutch municipality. In their experiments, the au-

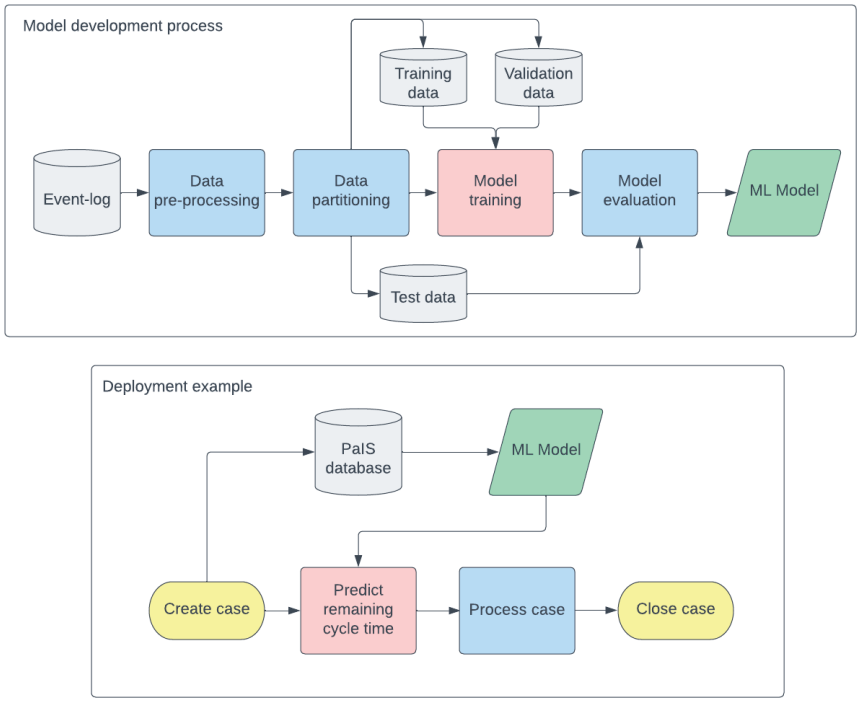


Figure 1.3: Predictive process monitoring example.

thors demonstrated the performance of using: 1) Only the attributes of cases which where static 2) Only the observed activities within the case, 3) Only the current cycle time at the time of prediction, 4) A combination of the previous three variants. Interestingly, the results showed that the static case attributes (1) were most the informative of the four variants, especially in the early part of the cases, where the second-best approach was the combined model (4). This initial work led to multiple approaches to the prediction of remaining cycle (also known as throughput time in this literature). Examples are *query catalogs*, which are based on conditional averages in historical data (Bolt and Sepúlveda, 2014), or hybrid approaches such as *predictive clustering trees* combined with *finite state machines* (Folino et al., 2013). A comprehensive review can be found in (Verenich et al., 2019). Common for the early work is that the approaches were not based on the most widely used Machine learning algorithms at the time (Hastie et al., 2001).

The work of (Evermann et al., 2016) presented a novel approach to predicting the next activity in a sequence, using a combination of Recurrent neural networks (Goodfellow et al., 2016) and techniques used in the field of natural language processing. Here, the authors took advantage of the similarity between a sequence of words or characters of varying length, and a sequence of activities within a case. Fol-

lowing this work, the authors in (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017) proposed an enhanced approach using Long short-term memory Recurrent neural networks (LSTM-RNN) (Hochreiter and Schmidhuber, 1997), capable of predicting the; time to next activity, type/class label of next activity, remaining sequence of activities (suffix) and remaining cycle time. In their approach, remaining cycle time was predicted at the sum of the predicted timestamps of the predicted suffix of a case. For remaining cycle time, their approach outperformed previous approaches in three of the four event-logs in which it was evaluated. One limitation to this approach was reduced performance on sequences with repeated patterns of the same activity.

In (Navarin et al., 2018), the authors improved the approach to remaining cycle time prediction proposed by (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017). This was achieved by changing the architecture of the LSTM-RNN, such that only predicted output was only a single value (the remaining cycle time), whilst also using all available attributes. This approach exceeded the performance on remaining cycle-time prediction achieved in (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017), whilst not being sensitive to sequences with repeated patterns of the same activity. The authors in (Camargo et al., 2019a) continued the work with LSTM-RNN prediction models by studying the influence of so-called embeddings, which is a n-dimensional projection of the input space. This technique enables the ability to generate a mapping between categorical inputs, which can be further used during training and prediction. Similar to (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017), the approach enabled prediction of time to next activity, type/class label of next activity, remaining sequence of activities (suffix) and remaining cycle time.

Two benchmark studies (Verenich et al., 2019; Rama-Maneiro et al., 2020) have have since found the approach in (Navarin et al., 2018) to perform the best for remaining cycle time prediction, evaluated across multiple domain data. For time to next activity, it was found in (Rama-Maneiro et al., 2020) that the approach of (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017) performed the best. For next activity prediction, clustering of event attributes combined with a LSTM-RNN (Hinkka et al., 2020) performed the best. Finally, the embedding approach of (Camargo et al., 2019a) proved to be best for suffix prediction. Common for all four prediction tasks is that the best performing solution is based on a Recurrent neural network.

Performance of prediction models

From the model evaluation perspective, the benchmark study of (Verenich et al., 2019) evaluated remaining cycle time prediction models based on two properties: 1) *Accuracy* performance, denoting the average performance across all prefixes, 2) *Earliness* performance, denoting the performance with respect to time. From the

latter perspective, the higher the accuracy at early time steps, the better earliness. To compare earliness performance, accuracy at different time steps thereby need to be measured. Going back to the three analytics types discussed in section 1.2.3, the earliness property of a case outcome or cycle time prediction would ensure the highest business value (as illustrated in Figure 1.2).

Another perspective on model evaluation was suggested by (Teinemaa et al., 2018b), who studied the *temporal stability* of case outcome predictions. The case outcome could in this case be compliance in terms of organizational goals, a fraudulent request, a deadline violation formulated as a binary outcome etc. The evaluation is motivated by reducing the volatility in classification models used for successive predictions: changing the conclusion multiple times will most likely cause ill treatment of a patient or customer or result in a lack of trust in the prediction model. The temporal stability is defined a measurement of the sensitivity to small changes in the input, in terms of predicted outcome. The results showed that LSTM-RNN (Hochreiter and Schmidhuber, 1997) and XGBoost (Chen and Guestrin, 2016) had the best temporal stability when using exponential smoothing, at the cost of accuracy performance.

As earliness is an important property in many business problems, especially in situations where there is a delay between the prediction and a possible action, this remains an important issue. Most studies evaluate a given approach from the aspect of both accuracy and earliness, by measuring the accuracy at different prefix lengths. However, no research has been made on optimizing the weights of a LSTM-RNN with respect to this objective. More specifically, as the loss function of a machine learning algorithm can be modified to prioritize one or more goals (as in (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017)), the loss function could also be modified to influence earliness or other performance aspects as well. Currently, only the LSTM-RNN approaches that focus on the prediction of multiple outputs using a single architecture (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017; Camargo et al., 2020) uses customized loss functions. As found by (Rama-Maneiro et al., 2020), the rest of the deep learning-based literature uses the L1 loss for cycle time prediction. Altering the loss functions is thereby an avenue of predictive process monitoring that have not yet been fully explored.

1.2.5 Prescriptive process monitoring

As described in section 1.2.3, the stream of literature related to prescriptive analytics within business process management, is commonly referred to as *prescriptive process monitoring*. The main goal here is to improve the operation of a business process, using predictive process monitoring for event prediction, and most often a secondary component to prescribe/recommend *treatments* to open cases that can prevent an unfavorable outcome. For instance, interventions such as calling a customer to retrieve missing information or changing to a faster supplier might be

performed to reduce cycle time (Bozorgi et al., 2021). As argued in (Teinemaa et al., 2018a), performing these interventions comes at a cost, and the authors therefore propose a cost-benefit model determining whether to trigger an *alarm* and thereby an intervention on a given case. In Figure 1.4, two examples are illustrated. In example 1, cases are escalated to a senior agent with more experience if they are predicted to exceed the service level agreement deadline. In example two, a prediction model recommends which intervention to use, based on available and predicted case attributes.

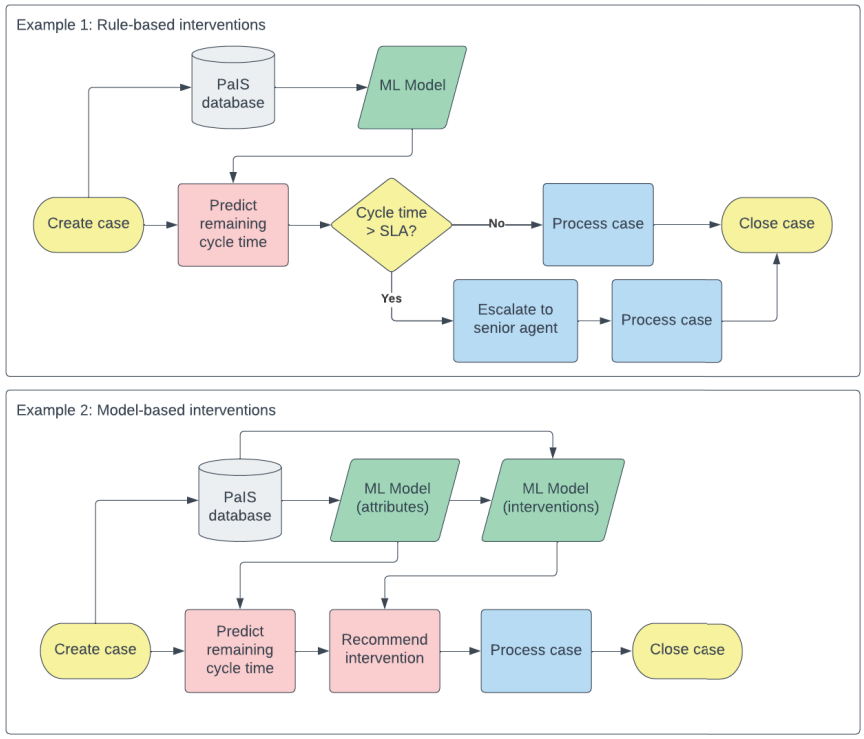


Figure 1.4: Prescriptive process monitoring example illustration.

The work of (Wibisono et al., 2015), which was based on the previous approach and data collected by (Nisafani et al., 2012), propose an automated resource-allocation approach based on a prescriptive Naive Bayes (Hastie et al., 2001) model. The experiments are performed using an agent-based simulation model (Railsback et al., 2006), calibrated from observed distributions in the case process (an Indonesian drivers license application process). At *runtime* (while the process is active), the model actively recommends which resource (police officer) from the resource pool to use, based on the conditional distribution of processing time for each resource on each previously observed activity type. While not accounting for fatigue, this

approach does improve the average cycle time.

Another approach can be found in (Thomas et al., 2017), where the authors use suffix prediction using Cascade neural networks, in order to identify case variants (unique sequences of activities) where a specific treatment is needed, in order to avoid a later *immediate* intervention. The authors do however not perform any simulation or field experiments, but rather evaluate the performance of their prediction model on historical data from emergency health records. The authors of (Teinemaa et al., 2018a), propose an alarm-based prescriptive method which calibrates the threshold of a predicted undesired outcome (referred to as an alarm), based on a cost model. The proposed framework then automatically finds a threshold for when to raise an alarm, given the cost model. In the experiments, the authors then conduct a return on investment analysis, in order to determine when a fictive prescriptive process monitoring system is profitable.

A recent stream of literature proposes conditional average treatment effect (CATE) estimate the effectiveness of a given prescriptive method, using historical data alone. For instance, (Bozorgi et al., 2021) propose a prescriptive process monitoring method for the reduction of cycle time, assuming that certain interventions can be performed at a given cost, in order to reduce cycle time. An example mentioned in the study is to proactively call and ask a customer for missing information, when this is delaying the process. The authors use a Orthogonal random forest to estimate the conditional average treatment effect of the assumed interventions, and based on a cost model, aim to improve cycle time when keeping costs at a given level.

Similarly, (Shoush and Dumas, 2022) propose a framework using two Machine learning models: 1) For estimating the likelihood of a undesired outcome, 2) For estimating the conditional average treatment effect. The focus of the study is resource allocation, where a resource is allocated to a task based on a cost model (which determines if and when to perform the intervention). The treatment effect of the model-based interventions is thereby estimated on the historical data. A weakness of the CATE method is as pointed out by the authors in (Bozorgi et al., 2021), that the treatment assignment should be independent of the potential outcome (randomly assigned or free of selection bias). A violation to this condition will bias the causal model and thereby invalidate the results. Arguably, knowledge about context of the business process and the assignment of treatments would help to reduce this risk, however, as argued by the authors of (Bozorgi et al., 2021), the only way to be sure is an experimental design with random assignment. However, as both the aforementioned studies (Bozorgi et al., 2021; Shoush and Dumas, 2022) uses publicly available benchmark data to conduct the experiments, the random assignment of treatments cannot be verified unless disclosed explicitly in the documentation of the data. While the proposed frameworks might work in theory, their estimated treatment effects might be biased by unobserved confounding variables which again

affects the internal validity.

The review in (Kubrak et al., 2022) show that most of the research in prescriptive process monitoring focus on optimizing efficiency-related metrics such as the cycle-time of an individual case, or the average cycle time of the business process. *Soft* key performance indicators related Customer relationship management have currently not been studied (see quality dimension in section 1.2.2). Another limitation in the current stream of literature, is the focus on publicly available benchmark data compared to case studies or field experiments. In fact, only 1 of the 37 studies retrieved in (Kubrak et al., 2022) based their findings on field experiments.

1.2.6 Business process simulation models

Traditionally, business process simulation models were derived using observation, interviews and analysis of documents related to the process being simulated (Hlupic and Robinson, 1998). A business process can be represented through a discrete event simulation model (see (Fishman, 2001)), consisting of 4 main components (van der Aalst, 2015): 1) The arrival process, which defines cases coming into the system 2) The control flow, which denote the sequence of activities, 3) Resources who perform the activities, and finally 4) Duration of activities.

For the generation of business process simulation models, the field of process mining (van der Aalst, 2016) have played a key role as it has enabled automated construction of the control flow model used for discrete event simulation (van der Aalst et al., 2004). Furthermore, as process aware information systems have become more widely used, researchers no longer need to rely on observation in order to estimate population parameters of e.g. duration distributions, as these can be estimated directly from event-log data produced by these systems. Within Process mining, a lot of research have therefore been made on the generation of accurate simulation models (digital twins) that represent the real world process as good as possible. The aim with these models is most often to generate *what if* scenario analysis, understanding the impact of certain changes to the business process by simulating them within a *digital twin* (van der Aalst, 2015).

Both commercial software such as ARENA (Altiok and Melamed, 2007), as well as open source alternatives such as ProM (van Dongen et al., 2005) exist for the generation of a business process simulation model. In recent years, the research within this field have focused on generating as realistic simulation models as possible. For instance, (Szimanski et al., 2013) proposed the combination of agent-based simulation and process mining, in order to create a hierarchical representation of the business process, where interactions between agents would be represented (as messages) represented. The authors of (López-Pintado and Dumas, 2022) studied the impact of the assumption that resources have identical availability calendars and performance. More specifically, they proposed a method to automatically differentiate both of these aspects in a simulation framework. The results showed that

these two aspects did indeed affect the overall accuracy of the simulation model.

A significant focus in the literature has also been on the automated end-to-end generation of simulation models, such as in (Mesabbah and McKeever, 2018; Camargo et al., 2019b, 2020). Based on the requirements of a simulation model, when used for *what if* analysis, this stream of literature generally aim at representing the reality as accurately as possible. To support the research of prescriptive process monitoring, these models are thereby ideal candidates, as the prescriptive process monitoring system can be evaluated from a digital twin of the process, instead of conducting a field experiment.

However, for predictive process monitoring, where hypotheses might be related to the impact of certain characteristics of the data generating process, these simulation approaches might be less ideal. To test a hypothesis related to model prediction performance given certain types of control flows or duration distributions, a single process model calibrated from a single real world process will lead to low external validity. Instead, multiple control flows and duration distributions sharing the same characteristics (that is to be tested) will be needed. For instance, in (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017) the authors found that their model performed worse in 1 of 3 event logs, and found that this event-log had sequences with repeated events. In this case, this particular hypothesis could have been tested with a purely theoretical simulation framework.

1.2.7 Concept drift

When a predictive or prescriptive process monitoring system based on Machine learning is implemented in a business process, the underlying distributional assumptions might not hold. When developing a Machine learning model, the underlying assumption is that the data used for model development comes from the same population as it will be implemented in (i.e., the business process itself). Even though the data might be collected from the same business process as it is to be implemented in, this assumption does not always hold, as the population might suffer from concept drift (Tsymbal, 2004).

Concept drift is a phenomenon where changes happen to a distribution over time. This can happen in multiple patterns over time (see (Gama et al., 2013)), and influence the strength or nature of relationships in the data which are modelled using Machine learning in Predictive and Prescriptive process monitoring. When the relationships in the population changes after a model has been developed, its accuracy will also decrease when used in the *new* population. This will thereby lead to a lower quality in the decision support provided by the predictive or prescriptive system, ultimately causing more harm than good. The process mining community have been studying these phenomena for business processes, and an overview of types of process changes can be found in (Bose et al., 2013).

To reduce problems occurring from these phenomena, the Machine learning model must be *adapted* to the new population by being re-trained using new data. This is commonly referred to as *drift adaptation* (Gama et al., 2013), and can be performed in two ways: 1) Blind adaptation, 2) Informed adaptation. Blind adaptation refers to retraining the model as soon as new data is available, whereas informed adaptation requires a detection system (Bose et al., 2011) to trigger the re-training once drift is detected. Blind adaptation can be very resource-intensive, especially for the LSTM-RNN-based models commonly used in predictive process monitoring (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017; Navarin et al., 2018).

In their experiments with predictive process monitoring models, the authors of (Maisenbacher and Weidlich, 2017) find that Hoeffding Trees perform particularly well for blind adaptation (compared to Naive Bayes and Perceptron), where Adaptive Hoeffding Option Trees perform best for informed adaptation (compared to Adaptive Hoeffding Trees and Single Drift algorithm). Similarly, (Baier et al., 2020) studied the performance gains in classification accuracy of a predictive process monitoring system by using informed adaptation. The authors found that the combination of incremental learning and drift detection (updating an existing model when drift is detected) yielded a performance increase of 28% in classification accuracy. In this study, the authors used a combination of drift detection and data selection methods: Page-Hinkley drift detection (Page, 1954) and ADWIN (Bifet and Gavalda, 2007) data window selection. Focusing on blind adaptation strategies alone, the authors of (Márquez-Chamorro et al., 2022) evaluate the performance of a Random Forest (Hastie et al., 2001) model used for predictive process monitoring in event-log data with concept drift. The authors proposed five strategies: Baseline, Cumulative, Non-cumulative, Ensemble, Sampling and Drift, where the Ensemble strategy was found to be the most effective in terms of both computational cost and performance over time.

A different stream of literature in the so-called AutoML community (Feurer and Hutter, 2019) focus on another aspect of this problem: Automating the model development process itself, and more specifically to do this in a cost-efficient manner. As the space of candidate settings (hyper parameters) can become very large, this stream of literature rely on a variety of methods to automate model development (Feurer and Hutter, 2019). Within this literature, it is therefore common to use a computationally efficient type of optimization algorithms known as Metaheuristics (Blum and Roli, 2003). One example from the predictive process monitoring literature is the work of (Francescomarino et al., 2018), where a framework for automated development of a predictive process monitoring system is proposed, based on a Genetic Algorithm (which is a Metaheuristic optimization algorithm). As the many other AutoML approaches (Muñoz et al., 2015; Hutter et al., 2013; Elsken et al., 2019; Maclaurin et al., 2015), this framework does not address the issue of concept drift.

In relation to adapting computationally demanding methods such as the LSTM-RNN (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017; Navarin et al., 2018), the methodology used within the AutoML literature, especially the stream based on Metaheuristics thereby appear as promising solutions for the *re-training* aspect itself. However, as discussed earlier, blind retraining is in itself computationally demanding. An overview of the existing approaches using Metaheuristics for automated drift adaptation could help advance this stream of research within predictive process monitoring.

2. Research questions

This section will present and motivate the individual research questions guiding the work of this thesis. The research questions aim to solve problems in individual areas, which combined will help achieve the overall research goal of this project, which is to:

- Improve methods for predictive and prescriptive process monitoring.

To achieve this, research work in four related areas has been conducted: 1) Improvements in the ability of predictive process monitoring to act as early warning systems, 2) Contributions to the evaluation of predictive process monitoring methods via synthetic data, 3) Suggesting a new method for case queue management via prescriptive process monitoring, 4) Providing an overview of the literature on automatic concept drift adaptation, suitable for methods in predictive process monitoring. In the following, the research questions of each of the four studies will be motivated and presented.

Study 1 will focus on predictive process monitoring. In the current stream of literature, effective models have been proposed for the prediction of the remaining cycle time of open cases in a business process. More specifically, the approach of Navarin et al., 2018 has been found to perform the best for this task across multiple proposed approaches in the literature (Verenich et al., 2019; Rama-Maneiro et al., 2020). Both in terms of the average prediction accuracy, as well as the timing of the error (referred to as earliness), this approach proves to be best across a set of publicly available benchmark event-log data. As the remaining cycle time of an open case is monotonically decreasing as the case progresses toward completion, a sequence of remaining cycle predictions would be expected to behave in the same manner. From a prescriptive process monitoring perspective, a predicted remaining cycle time that increases as time progresses (and is thereby not stable), poses a risk of prescribing the wrong intervention. However, it is currently unknown to which degree models such as the one proposed by Navarin et al., 2018 behave in this respect.

Common for the approaches for remaining cycle time prediction based on Machine learning, is the L1 (Mean Absolute Error) loss function (Goodfellow et al., 2016),

which is known to be less sensitive to large differences in the values of the target variable (Verenich et al., 2019). This particular loss function optimizes the Machine learning towards the median of the target distribution and has thereby proven to result in the best average accuracy. However, this particular loss function does not account for the timing of the errors, and thereby also only optimizes the model from the accuracy performance perspective. As discussed in Section 1.2.3, the earlier a useful prediction can be made, the earlier a prescribed intervention can be made, and from the logic of Figure 1.2, the greater the business value. The effect of altering the loss function such that it is *penalized* for bad earliness performance (via temporal weighting of the errors) has currently not been studied.

Consequently, Study 1 will aim to understand the influence of temporally-weighted loss functions for remaining cycle time predictions. Furthermore, the performance will be assessed in terms of temporal consistency, which denotes a monotonically decreasing prediction of the remaining cycle time as the case progresses. The work is guided by the following research questions:

- **RQ1** How can early warning performance of remaining cycle time predictions be improved?
  - **RQ1.1:** How do temporally weighted loss functions influence the performance of LSTM-based remaining time prediction models?
  - **RQ1.2:** To which degree do the predictions generated by LSTM-based remaining time prediction models fulfil the criterion of temporal consistency?

Study 2 seeks to make an epistemological contribution to the field of predictive process monitoring. In previous studies such as Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017; Camargo et al., 2019a as well as Study 1 of this thesis, performance aspects of the proposed model or method have been limited by the data from which it was evaluated. More specifically, by using publicly available event-log data from conference contests (La Rosa and Soffer, 2013; Teniente and Weidlich, 2018) or previous studies (Mannhardt and Blinde, 2017; Mannhardt et al., 2015), the researcher does not have full control of the data-generating process. The data can be understood through descriptive analysis, but the process that generated the data cannot be modified once the data has been collected. This fact thereby limits the understanding of its influence on model performance, as one cannot generate other scenarios from this process unless changing the real-world process and sampling another event-log.

Monte-Carlo simulation is thereby an important tool that can bridge this gap in the understanding of the performance and sensitivity of predictive process monitoring methods. Using simulated event-log data, not only data generating characteristics within a single business process can be controlled, but these changes can also be understood across multiple simulated business processes. As an example, the

authors in Camargo et al., 2019a seek to understand the influence of process complexity and variability on the performance of their proposed method. In this case, the authors selected 9 publicly available event-logs and classified them according to their degree of complexity or variability. As the number of publicly available event-logs that match the particular hypothesis a researcher might want to test is limited, this approach has some disadvantages: 1) Data collection and analysis is time-consuming, 2) There is no guarantee that the needed number of event-logs with the exact characteristics of the hypothesis is available. In the example of Camargo et al., 2019a, this led to an uneven distribution of logs within and across the two factors (complexity and variability). This can limit the results, as some scenarios are over-represented compared to others, and interaction effects might thereby be biased. In this example, synthetic event-logs could have been generated with systematical changes to process complexity and variability via an experimental design. This would have provided the ability to estimate interaction effects, as well as to control for the variation across multiple processes.

Study 2 thereby seeks to understand the capabilities of current simulation frameworks that can generate event-log data, and on the basis of this knowledge to contribute with a framework that covers the capabilities not offered in existing frameworks. Study 2 is thereby guided by the following research questions:

- **RQ2** How can the evaluation of predictive process monitoring methods via synthetic data be improved?
  - **RQ2.1:** To which extend does current business process simulation frameworks support the requirements for model robustness assessment in Predictive process monitoring?
  - **RQ2.2:** How can the limitations in current simulation frameworks be addressed within a new framework?

Study 3 will focus on the area of prescriptive process monitoring. In this stream of literature, the focus is mainly on efficiency-related measures such as costs (Teinemaa et al., 2018a), cycle time (Wibisono et al., 2015), or a combination of the two (Bozorgi et al., 2021). Based on the literature review of Kubrak et al., 2022, the majority of the work in this literature focuses on process improvement from the resource or control flow perspective. As the literature in prescriptive process monitoring predominantly relies on event-log data, aspects not recorded in this type of data are largely overlooked. One such example is case queue management, which by definition happens before the initiation of a case, and is thereby unobserved in most event-logs. In (Wibisono et al., 2015), the task queue from the perspective of the resource/agent in a driver's license application process was improved using a prescriptive allocation of agents to tasks. However, the prioritization of the case queue before initiation was not in the scope of this study.

Previous literature in operations research has studied case queue prioritization based

on predicted cycle time. An example is Tan et al., 2012 where the authors utilize the shortest remaining time first (SRTF) discipline in a case study of a hospital emergency department. Other approaches such as Wang et al., 2020 study the longest remaining time first (LRTF) discipline based on predicted cycle time in the context of edge computing server allocation. Similar to the literature on prescriptive process monitoring, the utility function is most often related to time or costs. Currently, no work in either of these streams of literature focuses on *quality* performance such as customer satisfaction or loyalty. As customer loyalty is a central construct in relationship-oriented approaches to marketing and service management (Hallowell, 1996; Kumar and Shah, 2004; Sheth and Parvatiyar, 1995), this is arguably an important aspect of process performance: The more loyal a customer, the higher their lifetime value. The more loyal customers, the higher a company's expected future profitability (Blattberg et al., 2008).

Consequently, Study 3 will present a case study within the customer service process of a European internet and telecommunications provider. Focusing on a *quality* performance aspect, namely customer loyalty (measured by the Net promoter score (Reichheld, 2003)), Study 3 will seek to improve process performance via prescriptive queue prioritization. Using data provided by the case company, an agent-based simulation model of the customer service process will be generated in order to answer the following research questions:

- **RQ3** How can prescriptive queue management improve *soft* performance measures such as customer loyalty scores?
  - **RQ3.1:** What are the distribution parameters of the key process components needed to generate an agent-based simulation model of the service process in the case company?
  - **RQ3.2:** How does the number of agents influence the queue waiting time in the simulation period under different prioritisation schemes?
  - **RQ3.3:** What are the effects of the four queue disciplines on overall process performance?
  - **RQ3.4:** What are the effects of the proposed loyalty-based queue discipline on the simulated net promoter score?

Study 4 will focus on the problem of concept drift, briefly discussed in the previous chapter (see Tsymbal, 2004; Gama et al., 2013 for an in-depth description). As predictive and prescriptive process monitoring systems are intended to support business processes reliably on a daily basis, the problem of concept drift adaptation is important to address. As concept drift is not only limited to business processes and has been studied across many other fields (Žliobaitė et al., 2016), Study 4 focuses on the problem of drift adaptation of Machine learning models in any operational setting. Current work on drift adaptation within the predictive process monitoring community (Maisenbacher and Weidlich, 2017) mainly focuses on computationally

efficient methods such as Hoeffding Trees (Domingos and Hulten, 2000) and Random Forests (Breiman, 2001a), whereas cost-effective drift adaptation methods for more demanding models such as the LSTM-RNN (Navarin et al., 2018) has not yet been studied.

Study 4 thereby focuses on a subset of drift adaptation methods which are based on so-called Metaheuristics. These algorithms are known to be computationally efficient for optimization problems in the automated development of Machine learning models (AutoML) (Feurer and Hutter, 2019; Francescomarino et al., 2018). As drift-adaptation can theoretically involve any task in model development (Chapman et al., 2000), Study 4 aims to understand the current applications of Metaheuristics for drift adaptation across fields. As this goal overlaps multiple theoretical frameworks, the approaches will be analyzed with respect to relevant theory within AutoML (Feurer and Hutter, 2019), Concept drift (Tsymbal, 2004; Gama et al., 2013) and Metaheuristics (Blum and Roli, 2003). To achieve this, Study 4 will answer the following research questions:

- **RQ4** How can metaheuristics aid machine learning systems in automatic adaptation in settings with concept drift?
  - **RQ4.1** Which types of metaheuristics have been utilized for automated adaptation to concept drift?
  - **RQ4.2** What characterize the application area of the use-cases?
  - **RQ4.3** How does the use-cases utilize metaheuristics for concept drift adaptation?
  - **RQ4.4** Which forms of concept drift were investigated?
  - **RQ4.5** How was the proposed use-cases evaluated?
  - **RQ4.6** What are the chronological trends in the found literature?

To help answer this primary research question, a set of six secondary research questions is proposed. The goal of the set of secondary research questions is to get a deeper understanding of the context wherein the algorithms are used: Which tasks they are used for, and in what context. Furthermore, the study aims to create an overview of the evaluation of the proposed methods: Which types and patterns of drift, as well as which evaluation method that was used.

3. Methodology

As Machine learning is a core element in this thesis, this chapter will start with a discussion of the pros and cons of typical methodology used in Machine learning research. Next, the overall research framework will be presented, and lastly, the chosen methodology in each of the four individual papers will be discussed.

3.1 Methodology in Machine learning research

In the classical paper; *Statistical modeling: The two cultures* (Breiman, 2001b), the author describes two fundamentally different approaches to the evaluation of statistical prediction models. The first approach is based on a *data model*, where evaluation of a given approach is based on Monte-Carlo simulation experiments (Rubinstein and Kroese, 2016) and goodness-of-fit tests. The second approach is based on *data sets* sourced from the setting wherein the prediction model is intended. The first approach has the advantage that it is transparent, but the disadvantage that it relies heavily on theory and assumptions about the data generating process, which might not represent reality. The transparency of this approach strengthens the internal validity, as the phenomena to be tested is specified by the researcher. However, from an ontological viewpoint, the phenomena must be observable from data in the first place.

In comparison, the second approach treat the data generating process as *unknown*, and rely on a combination of out-of-sample validation and accuracy measurements (Hastie et al., 2001). A clear advantage of this approach is the ecological validity, as models are evaluated within the environment in which they are intended to be used. From a *Humean* perspective (Anjum and Mumford, 2018), the external validity of such experiments simply rely on the number of representative settings (data sets) wherein the prediction model is evaluated. The withdrawal from clearly formulated assumptions about the data generating process, does however limit the knowledge produced with such experiments.

Some areas of Machine learning research try to take the *middle ground* between these two approaches (Tibshirani, 1996; Efron et al., 2004; Zou and Hastie, 2005; Bradley

and Henseler, 2007), combining the evaluation of a new approach by *benchmarking* on publicly available data sets used in previous research (commonly referred to as benchmark data) alongside Monte-Carlo experiments with clear assumptions. This approach have also been used in some parts of the literature stream of *deep learning* which focuses on Deep Neural Networks (Goodfellow et al., 2016), for instance in the first proposal of the Long Short-term Memory Recurrent Neural Network (Hochreiter and Schmidhuber, 1997), which is a central topic in this thesis. However, in recent years this approach have become less popular, especially in the field of Predictive Process Monitoring, which mainly rely on the so-called benchmark data for evaluation of a given method (Verenich et al., 2019; Rama-Maneiro et al., 2020).

Another aspect of the two approaches to evaluation in Machine learning research is the *transparency* of the models themselves. Traditional models used within the statistical community rely on the ability to make *inference* and draw conclusions about relationships, given a set of assumptions. This approach is very much contrasted by the algorithmic modelling used in the Machine learning community, where prediction models have until recently (see (Lundberg and Lee, 2017)) been regarded as too complex to make clear inference about relationships.

Machine learning algorithms (Hastie et al., 2001) rely on a stochastic optimization process referred to as *training*, which results in a solution (a prediction model) that can generate predictions on the holdout sample with an given level of accuracy. Reproducing these results can be achieved by setting a given *seed value* (fixed value used to initialize the random number generator (Rubinstein and Kroese, 2016)). However, modern research on deep learning prediction models requires specific hardware in order to speed up the training process via parallel processing (Goodfellow et al., 2016). This, on the other hand, leads to hardware-dependent rounding-errors which in turn mean that results cannot be faithfully reproduced (Chen et al., 2022).

3.2 Overall research framework

To answer the research questions, a mix of methods have been used individually in each of the four studies, as well as combined in some studies. The primary methods used are Monte-Carlo simulation, descriptive analysis, predictive modelling and literature reviews. A graphical overview of the methodological framework can be seen from Figure 3.1. Individual discussions of the methods used in each of the four papers can be found in Section 3.3.

3.2.1 Data

As the thesis is centered around methods for process improvement via algorithms and predictive models, a key element is the data generated within business processes from the so-called Process-aware Information Systems. This data format is also

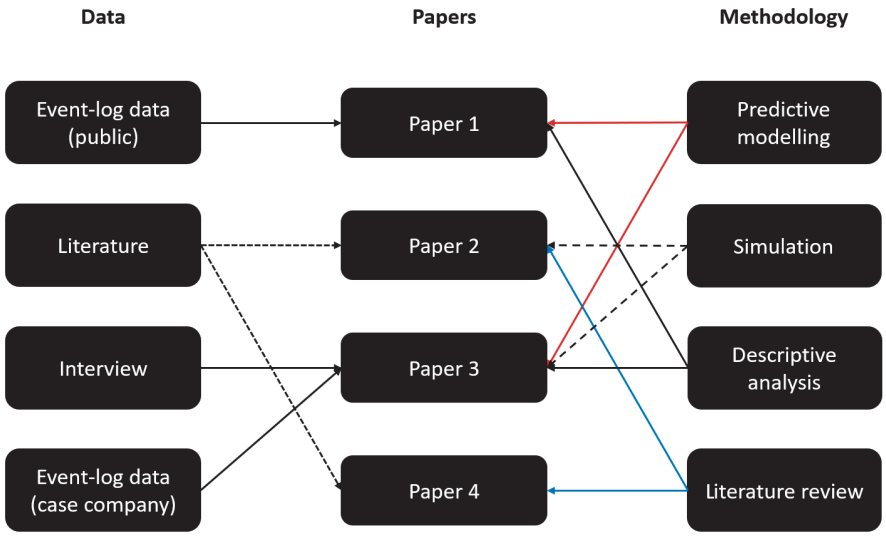


Figure 3.1: Methodological framework.

known as event-log data, where activities are logged in chronological order, and linked to a given case (sometimes referred to as a process instance). The event-log might carry information other than the activity and timestamp, and include other case-related attributes.

Papers 1 and 3 use event-log data as the primary data. Where paper 1 uses publicly available event-log data (most often used for benchmarking across studies), paper 3 uses event-log data from a case company. Where a higher number of data sets might be an advantage in terms of external validity, this also comes at a trade-off in terms of the amount of detail that can be put into the descriptive analysis. Papers 1 and 3 are from this aspect opposed to each other, as paper 1 has a descriptive analysis based on distributions and summary statistics alone, where paper 3 has a more in-depth analysis with process models and interviews with the case company. The collection of data was thereby adapted to the aim of the papers.

As paper 1 tests the effect of a modification to an existing predictive process monitoring method, it was essential to understand their differences in terms of the distribution of prefixes and case duration (cycle time). The data was sourced from four different domains to understand potential differences across four different processes. On the contrary, as paper 3 tests the effect of a queue-prioritization algorithm which is intended for customer service processes alone, a case study with a more detailed analysis of the process was needed in this case.

The event-log format has its limitations, as the lowest unit of observation is the activity, and other events that might happen in between the activities are not necessarily recorded in this data. One example is the event-log data used in paper

3, where aspects of human behavior between or alongside events are not recorded. For instance, the time at which an agent signs into the console and starts working, or when an agent decides to abandon or re-assign themselves to a case. The resource behavior can be inferred from the observed behavior in the event-log data, but as discussed, there is also behavior that is not observed from this type of data alone. As we were not granted access to data from systems logging detailed resource behavior, this limited the scope of paper 3, as non-preemptive queue disciplines had to be excluded. What also became apparent from the results of paper 1, was the limitations of the publicly available *benchmark* event-log data. As discussed earlier, hypotheses related to the data-generating process cannot be tested effectively using this data source, which motivated paper 2.

Papers 2 and 4 mainly use qualitative data in the form of existing literature. This is a very rich source of information, which enables a more detailed analysis of the phenomena in question. As paper 2 aims to improve the evaluation methods in predictive process monitoring by contributing with a new simulation framework, the capabilities of current frameworks were analyzed qualitatively. However, as some frameworks are presented in conference papers (where the level of detail is restricted by page numbers), this also limited the level of detail that could be understood about the distributions used, as well their availability in each of the existing frameworks.

Journals and conference proceedings also have different requirements in terms of the level of detail to be reported in a paper, which might also differ depending on the individual reviewers. For the literature review performed in paper 4, this limited the results as some of the included studies did not report specific aspects of the data and evaluation method used in their experiments.

3.2.2 Methods

As indicated by Figure 3.1, four different methodologies are used either separately or combined in each of the four papers of this thesis. Papers 1 and 3 use **predictive modelling**, which is based on partitioning the historical data into training, validation, and test sets (Hastie et al., 2001). The training set is used for *learning*, while the validation set is used for hyper-parameter optimization, which is the process wherein the optimal settings of the Machine learning algorithm are found (Goodfellow et al., 2016). The test set is used for the evaluation of the final model. In paper 1, this is the sample from which the results are reported, while this sample was not used in paper 3, as the results were based on simulation alone.

Papers 2 and 3 use **Monte-Carlo simulation** (Rubinstein and Kroese, 2016) in two different ways: Paper 2 uses simulation to test and demonstrate the characteristics of the event-log data generated by the simulation framework itself, whereas paper 3 uses a simulation of an existing process to estimate the impact of different interventions (queue disciplines). The two approaches are also different in terms of

the nature of the simulation model, as paper 2 uses a *parametric* simulation model (i.e. the data generating process is purely specified from parameterized distribution functions). Paper 3 uses an *agent-based* simulation model (Crooks and Heppenstall, 2011) with central elements such as case arrivals represented by parametric distributions calibrated from the primary data. Using an alternative method in paper 3, such as field experiments would have been both costly and risky to the case company, as the real process would have been manipulated, and real customers thereby would have been affected. As the number of agents was altered in the study, this would have led to sub-optimal process conditions and thereby unnecessary negative customer experiences. In this case, a digital twin of the existing process is a less risky approach to evaluating a prescriptive method before field testing.

Descriptive analysis is mainly performed in papers 1 and 3, as a way to understand the nature of the primary data. In paper 3, a descriptive analysis was performed to estimate distribution parameters of the customer service process in the case company, based on provided event-log data. The descriptive analysis thereby included modeling of the conditional distributions of key components such as case arrivals, activity durations, activity sequences, and the conditional loyalty response. These relationships were thereby used to calibrate the simulation model to the real process. In paper 1, the descriptive analysis primarily included summary statistics and prefix distribution plots, which is an illustration of the frequency of cases with events at given sequence lengths (prefixes). These were important to understand the general difference between the four benchmark even-logs used to evaluate the proposed loss functions.

The **literature review** methodology was used in papers 2 and 4, where the literature review in paper 2 was a preliminary part of the full study. Both papers used the method of semi-systematic literature review (Snyder, 2019), combined with qualitative analysis of the results. Where paper 4 used an incremental search strategy of 3 queries, paper 2 only used a single query to retrieve the results needed. For paper 4 the search area was wide and included multiple fields, whereas paper 2 only included the literature within the so-called process mining community, which uses the term *event-log* for data originating from PaIS. The choice of literature as a method for paper 2 was to understand the current contributions, and establish key criteria for further contributing via a new simulation framework.

3.3 Individual discussion of used methodology

In the following, the methodological choices made in order to answer the research questions in each of the four papers will be discussed in further detail.

3.3.1 Paper 1

The first paper: *Remaining throughput time prediction: Temporal loss functions and prediction consistency*, proposes three alternative loss functions for improving the so-called *earliness* performance, based on the regular L1-loss. To test the hypothesis that temporal weighting of the loss function can have an effect on model earliness, an experimental design approach is pursued. As the training of deep neural networks is based on stochastic optimization, the study tries to compensate for the statistical uncertainty by repeating the experiment 10 times. The aforementioned number of repeated experiments is relatively low; however, compared to similar studies (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017; Navarin et al., 2018; Verenich et al., 2019; Evermann et al., 2016) (who do not repeat their experiments), the statistical uncertainty of the results will be lower. In relation to the external validity of the results, the study uses four different datasets from different domains, with different characteristics (number of observations, trace lengths, duration distributions, etc.). In related literature proposing a method, two to three data sets have been used (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa, 2017; Navarin et al., 2018; Evermann et al., 2016), whereas survey/benchmark studies generally use more datasets (Verenich et al., 2019; Rama-Maneiro et al., 2020).

In relation to internal validity, it is common practice to use a temporal split of the data, partitioning it into two periods where the first is used for training and validation, and the remaining for testing. This approach does to some extent also improve the ecological validity, as practitioners cannot *freeze* time, and need to evaluate a model on cases arriving in the future. However, some studies (Verenich et al., 2019; Teinemaa et al., 2018b) include cases that span both the train and test periods to be used for training, in order to avoid dropping a significant amount of cases. An alternative to this approach would be what is referred to as *censoring* in the literature on survival analysis (Tutz, Schmid, et al., 2016). Censoring would in this case mean dropping cases that do not finish within either of the two periods.

Comparing these two approaches, there is a trade-off between internal and external validity, as censoring would improve internal validity and ensure that observations within each temporal split (train/test) do in fact represent the temporal distribution within that period, and not a mixture of both periods. Where, on the other hand, this approach will make results less comparable with similar studies not using censoring. As the focus of this study was to compare four loss functions for the same model type, and not across other model types or previous studies, the internal validity was deemed most important. If future studies aim to reproduce or compare these results without censoring, the full source code is freely available online.

3.3.2 Paper 2

The second paper: *A parametric simulation framework for the generation of event-log data*, proposes a simulation framework in the Python programming language

for the generation of synthetic event-log data, using well-known parametric distributions. The methodology of this paper is based on an initial semi-systematic literature review (Snyder, 2019) of a subset of the existing simulation frameworks.

As the aim of the framework was to contribute to future research in predictive process monitoring, only existing open source simulation frameworks in the Python programming language were included in the review. The emphasis of open source was such that future research could enable modifications to the presented framework, where the Python language was due to it being the most commonly used language in Predictive process monitoring. The selected frameworks were qualitatively compared in terms of: 1) Their general approach to simulation, 2) which components they include, 3) how they represent the control flow, and 4) which parametric distributions they used.

Since the proposed framework covers a set of features not supported by the existing frameworks found, a quantitative performance comparison with the existing frameworks was not possible. The main source of comparison is thereby the qualitative analysis in the literature review of the study. To demonstrate the capabilities of the framework, a set of simulation experiments were performed in addition to a detailed documentation of the implementation.

3.3.3 Paper 3

In the third paper: *Customer-service queuing based on predicted loyalty outcomes*, a prescriptive queue prioritization algorithm is proposed. The goal of the algorithm was to prioritize customers in the queue, in a manner that should increase the Net promoter score (Reichheld, 2003) on an organizational level.

The proposed algorithm was evaluated via a case study of an anonymous Scandinavian Internet and Telecommunications provider. The primary data was interviews with the case company, in order to understand the context of the process, as well as event-log data from the process itself. The event-log data was used to calibrate an agent-based simulation model, such that costly real-world experiments would not need to be performed to understand the pros and cons of the proposed algorithm.

In terms of ecological validity, certain aspects of the simulation model were manipulated, whereas others were kept true to the original process. The manipulated factors were the queue discipline, as this was the subject of comparison, and the number of agents, as this greatly influences the performance of any queue discipline. Furthermore, the data provided by the case company only included cases that were based on incoming emails from customers, whereas the real process also included cases created via phone calls.

All other aspects of the process were kept as close to the real process as possible, via a descriptive analysis of the provided event-log data. These results were used to calibrate the parameters of an agent-based simulation model. Monte-Carlo simula-

tions were then performed within the time span of the provided event-log data, such that the variation in the case arrivals would follow the same seasonal patterns as observed in the real process. The experiments were repeated 100 times to account for statistical uncertainty.

In the simulation experiments, the proposed approach (NPS) was compared to three other queue disciplines: First-come First Served (FCFS), Longest Remaining Time First (LRTF), and Shortest Remaining Time First (SRTF). The current discipline in the case company was FCFS, but as predicted cycle time was available at case arrival, LRTF and SRTF were included to understand performance differences between them and NPS. Other disciplines such as Service In Random Order (SIRO) or Last-Come First Served (LCFS) could also have been included; however, as these tend to lead to worse average cycle time than FCFS, they were not included.

The Net promoter score has been criticized in terms of its inability to represent the negative word of mouth or dissatisfied customers, who might not respond to a survey and thereby not be represented (East et al., 2011). Furthermore, a detailed investigation in Keiningham et al., 2007 found that the Net promoter score was inferior to other loyalty measures for predicting company growth, contradicting the claims by its inventor (Reichheld, 2003).

As the Net promoter score is actively used in the case company, we had no possibility to implement other loyalty metrics such as the ones compared in (Keiningham et al., 2007). From our conversations with the case company, it has also been clear that the Net promoter score was not used as a predictor for company growth, but rather as a measure of the customer experience. In terms of the criticism of dissatisfied customers who would not get measured (as they might not reply), we tried to circumvent this by including only cases that received a response after case completion. On the other hand, this reduces the external validity of the results, as the simulation results would not represent the full population. However, as we could not model the conditional distribution of the Net promoter score for cases without a response, we had to exclude these cases from the simulation model.

3.3.4 Paper 4

The fourth paper: *Automating model management: A survey on metaheuristics for concept-drift adaptation* is a review of literature across multiple fields, using a particular family of algorithms for automatically re-training Machine learning models in situations with concept drift. In the context of this thesis, the aim is to create an overview of the existing frameworks as well as the methodologies used to evaluate them, as this can help future research on drift adaptation in predictive process monitoring.

With these aims, a *semi-systematic* (Snyder, 2019) literature review is most appropriate, as this provides an overview, rather than systematically comparing results

across studies. This would have been problematic for multiple reasons: 1) The overlap of data used in the found studies is minimal, 2) The overlap of types and drift patterns is minimal, and 3) The method of evaluation (partitioning and metrics) vary across the studies. Furthermore, the source code is rarely provided open source, which would be needed if one were to re-create the experiments and perform a benchmark study such as Verenich et al., 2019.

The search methodology follows the general guidelines in Kitchenham, 2004 in terms of the documentation of the literature search and selection process. However, the selected literature only includes peer-reviewed publications with more than 2 citations when more than 2.5 years old at the time of retrieval. These criteria were selected in order to ensure quality in the retrieved literature. However, this approach does have a few limitations as it: 1) Does not guarantee that the included literature has a given approach to validation of the results, 2) Can lead to the exclusion of research fitting the other criteria, but with few citations due to a low interest in the scientific community. As this study was performed by a single author, the risk of reliability issues in the inclusion decisions is also higher than for multiple authors, as discussed in Kitchenham, 2004.

The literature was qualitatively coded based on the research questions which focused on three general areas: 1) The proposed framework, 2) The conditions wherein it was evaluated, and 3) The methodology used for evaluation. The coding of the found literature was closed for areas where the theoretical framework was clearly defined in the background section of the study, and open in cases where it was impossible to assess the values before the literature had been retrieved. One example is the type of data that was used to evaluate the framework.

4. Main contributions

The contributions of the conducted research in this Ph.D. project can be summarized as follows: Paper 1 contributes with new knowledge about the performance of remaining cycle time prediction models from two perspectives: 1) altering the loss function, by adding a temporal decay leads to improvements in both earliness and accuracy performance, 2) remaining time predictions are often non-monotonic, and can in some cases change the direction of the previous prediction with a high magnitude. From the experiments, these cases tend to happen for longer cases with low support in the training data. Paper 2 contributes with a literature review of current open source business process simulation frameworks and their capabilities in terms of evaluating data-related hypotheses. Furthermore, a new open source simulation framework in Python is proposed, covering some of the areas not supported by current simulation frameworks. The proposed framework is designed to be used explicitly for data-related hypothesis testing of predictive process monitoring methods.

Paper 3 proposes a novel prescriptive approach to queue prioritization in customer service, focusing on improving customer loyalty. Using a calibrated simulation model, it is found that the proposed method does improve customer loyalty scores in situations with reduced capacity. This comes at the trade-off of longer average waiting times. Finally, paper 4 contributes with an overview of the existing literature on drift adaptation using Metaheuristics. The literature is analyzed based on the relevant theory of concept drift, as well as Metaheuristics and Automated Machine learning. The results show that multiple approaches based on Metaheuristics can be assessed alongside the current algorithms used in the literature on concept drift adaptation in predictive process monitoring.

In separate ways, the conducted research in the included studies contributes to the overall research goal of this thesis, which was to improve methods in predictive and prescriptive process monitoring. In the following, the individual contributions of each of the four papers will be further described.

4.1 Paper 1

The first objective of paper 1 was to investigate the impact of temporally weighted loss functions for remaining cycle time prediction using the approach proposed by (Navarin et al., 2018). A second objective was to understand to which degree the predictions made from these models follow the monotonically decreasing behavior of remaining time. To achieve the first objective, three loss functions with different functional forms were proposed. The loss functions were all variants of the standard $L1$ loss with different degrees of temporal decay added. To achieve the second objective, a new evaluation metric called Temporal Consistency (TC) was proposed. The TC measure the degree to which a prediction model *increases* the predicted remaining time between two discrete events (activities). The purpose of evaluating the model from this perspective is to understand the consistency in the direction of the predictions, as direction changes can be problematic when used for dynamic planning. As an example, volatile direction changes might lead to frequent changes in policy, which might in itself increase costs.

In relation to the first objective, the results showed that the added temporal decay did indeed improve earliness performance by up to 4% at the second observed event in one case. The results showed general improvements from the earliness perspective across all four evaluated event-logs; however, only with statistical significance in two of the four event-logs. As the three proposed loss functions had different curvatures, and one of the losses outperformed the others, the results suggest that the functional form (peak and slope) influences earliness performance. The results also hint at performance improvements in the accuracy domain, as the temporal loss functions have the lowest average error across all four event-logs. However, these differences were statistically insignificant.

In relation to the second objective, a contribution of this study was the further understanding of model performance through the Temporal Consistency metric. The results indicated that from this aspect, model performance worsened in situations where the support in the training data was low, i.e. after a particular number of discrete time steps. This led to *peaks* in the measured TC error, which for practitioners would mean that the predicted remaining time: 1) changes direction, and 2) by a significant magnitude.

Comparing the performance of the temporal loss functions from this aspect as well, the results showed that the temporal weighting led to worse performance. However, the difference between the baseline MAE and the proposed loss functions is only statistically significant in one of the four evaluated event-logs. The evidence of a trade-off between accuracy/earliness and temporal consistency is thereby present, but weak if based on these results alone. In addition to these results, a contribution to future research in this area is the open source implementation of the loss functions in the Tensorflow/Keras deep learning framework (Martın Abadi et al., 2015). The

source code used for training and evaluation is available online as reported in the paper.

4.2 Paper 2

Paper 2 aim to contribute to the epistemology of predictive process monitoring. More specifically, in relation to the evaluation of predictive process monitoring models. This is achieved by solving two objectives: The first objective is to understand the capabilities of current business process simulation models, via a literature review. This is limited to frameworks that are open source and programmed in the Python language. The motivation here is: 1) That the majority of the recent frameworks for predictive process monitoring are developed in Python, and 2) That a simulator should be transparent when used for scientific research. Furthermore, the gaps in the capabilities of existing frameworks are assessed in order to establish design criteria for a contribution in this area. The second objective is thereby to develop a simulation framework that offers features not currently covered by existing frameworks.

The results showed that only 4 of the 9 found frameworks have the capability to alter the distributions of a generated simulation model. The majority of the frameworks focus on generating a simulation model from existing data as automated as possible (with the least amount of input from the user). Moreover, the distributions used within the frameworks are often poorly documented, if reported at all. In the case of testing data-related hypotheses, transparency of the data-generating process is essential. A key requirement for a contribution was therefore a transparent data-generating process, with clearly documented distributions and algorithms. As the found simulation frameworks had an *empirical* focus (re-creating an existing process), rather than enabling the researcher to generate theoretical processes with different levels of complexity (as previously studied in (Camargo et al., 2019a)), control over this aspect of the simulation model became another key design requirement.

To contribute to future research in the field of predictive process monitoring, an open-source simulation framework developed in the Python language was thereby proposed. The framework is based on Markov chains and Higher-order Markov chains for the simulation of activity sequences (the control flow), while the duration distribution is based on the Hypo-exponential distribution. The choice of these distributions enables the framework to simulate event-logs generated from processes without memory (the probability of each activity is only dependent on the previous), as well as processes with different degrees of memory (the probability of each activity is dependent on the last k activities). Using the Hypo-exponential distribution enables the ability to simulate unique distribution shapes for not only each individual activity, but also the temporal order of the activity. Further capabilities of simulating stochastic offsets in activity duration, as well as deterministic offsets

for representing business/office hours, were added.

As other data-related hypotheses than those demonstrated in the paper might be relevant in future research, a key contribution of this work is thereby the open sourced code. Users might therefore either use the framework as reported in the paper, modify the currently used distributions or add capabilities to suit their specific needs. The current version of the framework includes the capability to generate experimental designs and can be easily modified to train and evaluate any Python-based Machine learning model from the generated event-log data.

4.3 Paper 3

Paper 3 proposes a novel approach to queue prioritization based on predicted loyalty scores. The proposed approach uses predicted cycle time to predict individual loyalty scores, in order to prioritize a segment of the customers waiting in the queue. The study was divided into two sub-studies: Study 3.1, which uses information and historical data from a customer service process within the case company (a European internet and telecommunications provider) to calibrate an agent-based simulation model. In Study 3.2, Monte Carlo experiments were performed from the generated simulation model, comparing four different queue disciplines: first-come first served (FCFS), shortest remaining time first (SRTF), longest remaining time first (LRTF), and finally, our proposed Net promoter score-based approach (NPS). As all approaches but the FCFS discipline suffer from starvation (some customers being stuck in the queue), we also simulated a scenario with a service level agreement (SLA) of 72 days. In this case, the queue scheduling algorithm was modified such that cases waiting more than 60 hours in the queue were put in front of the queue (ensuring that agents had 12 hours to initiate and process the case). In addition, we varied the number of agents available in the process.

A key contribution of this study is the finding that customer loyalty scores can be influenced by adaptive queue prioritization of customer service cases. In the experiments, we found that our NPS approach led to the best customer loyalty scores on the organizational level. This was closely followed by the $LRTF$-approach, which acted in a similar manner, using the same input (predicted remaining cycle time). The results indicated that the aforementioned two approaches brought down the average cycle time to a similar level, clearly outperforming $SRTF$ and $FCFS$ in this aspect. However, a clear trade-off was the increased average waiting time in the queue, where $FCFS$ consistently outperformed the remaining three approaches. The effects on loyalty scores, resolution time and waiting time were largest when the process was under-staffed, and vanished once staffing was large enough to eliminate the queue. Furthermore, when simulating the scenario with an SLA of 72 hours, all disciplines performed identically, as the queue quickly reverted to the $FCFS$ discipline.

A key takeaway is thereby that the proposed method is only effective in situations where optimal staffing cannot be achieved at all times, and thereby most appropriate for *lean* organizations or in periods with high service demand. As the proposed NPS approach is prone to starvation, the proposed SLA-modification could also be applied to ensure fairness. However, for the simulated process in the case study, the SLA of 72 days proved to be too strict. Optimal values for this parameter thereby needs to be calibrated from the individual setting.

A general finding is that even when the relationship between case cycle time and NPS-response after case closure is rather weak ($R^2 = 0.04$ in this case study), customer loyalty scores can still be significantly improved using our prescriptive approach. As cycle time predictions were based on incomplete information (seasonality indicators alone), being able to use case attributes such as case topic for these predictions might further improve performance of the disciplines based on these predictions. However, as this information was not available at the time a case entered the queue, this was not included in the prediction models.

4.4 Paper 4

Paper 4 seeks to contribute to the literature on concept drift adaptation with the aid of Metaheuristics. This is motivated by the fact that these computationally efficient algorithms are widely used for the automated development of regular Machine learning models (AutoML). However, this stream of literature does not address the problem of concept drift, and the aim of this study is thereby to understand how Metaheuristics are used in research with automated drift adaptation. As the current literature on concept drift adaptation in predictive process monitoring mainly uses streaming algorithms such as Hoeffding Trees (Domingos and Hulten, 2000), or lightweight Machine learning models such as Naive Bayes or Random Forests (Hastie et al., 2001), the issue of drift-adaptation for computationally demanding approaches such as the LSTM-RNN (Navarin et al., 2018) remains unsolved.

The results show that Metaheuristic algorithms are used across a broad spectrum of Machine learning problems. 6 out of 17 studies use Metaheuristics to adapt a Neural Network-based Machine learning model, whereof 2 are Recurrent Neural Networks. The found literature is classified based on the relevant theory of Automated Machine learning and Concept drift. The use-cases mentioned before apply the Metaheuristic in different steps of the model development cycle (see (Chapman et al., 2000)), and the tasks thereby vary from feature selection and hyper-parameter optimization to the so-called full-model search, which automates every step in model development. However, only 2 of the 6 mentioned studies use a drift detection method, whereas the rest rely on blind adaptation. In these two cases, the drift detection is based on exceeding a given threshold of error.

Of the found studies, 5 uses incremental learning and tree-based methods similar to

the approaches that have been previously studied for predictive process monitoring (Maisenbacher and Weidlich, 2017). The main difference between these approaches is the inclusion of Metaheuristics to aid in feature selection, data window selection, model management, or model training (model-level optimization as in (Karimi et al., 2012)). In the line of current work on drift adaptation in predictive process monitoring (Maisenbacher and Weidlich, 2017; Baier et al., 2020; Márquez-Chamorro et al., 2022), other tree-based methods from the found literature of this study might be relevant to evaluate alongside current approaches. For the adaptation of deep learning-based methods such as LSTM-RNN (Navarin et al., 2018), one approach based on Genetic Algorithms (GA) was found (Kumar and Batra, 2018). In this particular framework the GA is used to decrease the number of computational resources needed for hyper-parameter optimization by: 1) Selecting an optimal subset of the data to train from (based on current error), and 2) Guiding the configurations to be trained (through an evolutionary strategy).

More generally, the results of the analysis show that real-world data is most often used for the evaluation of concept drift adaptation. The drift type of these data sources is often unclear and seldom stated explicitly in the found studies. This is similar for the drift patterns, which are unknown for 10 of the 17 found studies. For the chronological trends in the found literature, a clear transition was found from automation of single tasks such as feature selection or hyper-parameter optimization towards full-model search in recent literature.

5. Reflection and discussion

Multiple new ideas for the improvement of predictive and prescriptive process monitoring have been presented in this thesis. However, when reflecting on the limitations of the results, multiple other directions could have been chosen to address the main research objective.

One example is the results of Study 1, which were limited by the flexibility or level of difference between the loss functions, as well as the data that was used to evaluate them. A different approach could thereby have been to use only one loss function, where the shape and peak of the temporal decay could have been controlled by parameters. The proposed simulation engine from Study 2 could also have been used to gain a deeper understanding of how data-related factors influence the performance differences between the loss functions. However, as Study 2 was motivated by the limitations of Study 1, this was not possible. Instead, this should therefore be addressed in future research. Instead of remaining cycle time prediction, outcome-oriented prediction such as the one proposed by Teinemaa et al., 2018a could have been used. However, the simplicity of the selected prediction problem and the utilized loss functions proved to be beneficial in this work. The achieved improvements in accuracy and earliness from using the temporal loss functions were also marginal; however, as found in the experiments of Study 3, even weak prediction models have an effect in the long run when used operationally.

Rather than proposing a new framework in Study 2 to aid in model evaluation, an existing framework could have been modified. However, as the design philosophy of the found frameworks was fundamentally different from the approach in Study 2, this would greatly reduce the benefits of such an approach. For its intended use, the proposed framework can ideally shed further light on data-related factors that can influence the performance of predictive process monitoring methods. Potential findings in this area might therefore also motivate new data encoding schemes or feature transformations to counter potential problems as was the case in Camargo et al., 2019a. A limitation of the proposed framework is, however, the lack of the ability to calibrate the distributions to those of an existing event-log. However, these capabilities are already offered by the existing frameworks found in Study 2.

If the proposed framework were to be used for the evaluation of case outcome predictions as discussed in relation to Study 1, the current formulation would be insufficient (depending on the formulation of the case outcome). For the use of next activity prediction, the application is also somewhat limited, as the activity sequence is exclusively dependent on Markov Chains and independent of the other simulation components. Conversely, the activity sequences and time components all influence the duration and waiting time distributions.

From the perspective of prescriptive process monitoring, the work in this thesis is limited to a single case study which is a strength in terms of ecological validity, but a weakness with regard to external validity. However, as there is an (albeit weak) relationship between cycle time and resulting customer loyalty scores, similar settings with this property might benefit from the proposed method. However, a fundamental downside of the proposed method (when applied without a service level), is the unfair treatment of customers. In its current form, the applicability of the proposed method thereby relies on a *safety-mechanism* to counter the issue of starvation. A further understanding of the relationship between the service level, number of agents, NPS-distribution, cycle time distribution, and arrival rate would have been beneficial if systematically varied in a separate study.

To follow up more directly on the work in Study 1, the work in Study 3 could have focused exclusively on cycle time prediction by studying the effect on process performance when using LSTM-RNN (Navarin et al., 2018) models for queue prioritization. In this way, the impact on business process performance from the error types investigated in Study 1, could have been further investigated from this case study. However, to evaluate such approaches, a more complex data-generating process than the one in Study 3 is arguably needed to leverage the benefit of the LSTM-RNN models. One way to achieve this could be by extending the agent-based simulation model with more complex sequential distributions for both activities and durations, similar to the approach proposed in Study 2.

An alternative to the current work on concept drift in Study 4 could also have been to conduct experiments where drift was introduced to the setting of Study 3. However, as the current simulation model is calibrated from data with seasonality patterns in the arrival rate of new cases, this is arguably already the case. This would in itself also not solve the problem of finding new methods for drift adaptation. Although limited by the search methodology used, the results of Study 4 revealed multiple methods suitable for further drift-related research in predictive process monitoring. For instance, current approaches do not address the problem of unsupervised learning with drift or drift-adaptation of deep learning-based methods such as the one in Study 1. The results of Study 4 thereby reveal multiple candidates for further research in this area. However, as these were not evaluated in this project, the state of drift adaptation in predictive process monitoring remains unchanged.

References

Agutter, C. (2020). *ITIL® 4 Essentials: Your essential guide for the ITIL 4 Foundation exam and beyond*. IT Governance Ltd.

Altiok, T. and Melamed, B. (2007). Chapter 2 - Discrete Event Simulation. In: *Simulation Modeling and Analysis with ARENA*. Ed. by T. Altiok and B. Melamed. Burlington: Academic Press, pp. 11–21. DOI: https://doi.org/10.1016/B978-012370523-5/50003-1.

Anjum, R. L. and Mumford, S. (2018). *Causation in science and the methods of scientific discovery*. Oxford University Press, USA.

Baier, L., Reimold, J., and Kühl, N. (2020). Handling concept drift for predictions in business process mining. In: *2020 IEEE 22nd Conference on Business Informatics (CBI)*. Vol. 1. IEEE, pp. 76–83.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, pp. 443–448.

Blattberg, R. C., Kim, B.-D., Neslin, S. A., Blattberg, R. C., Kim, B.-D., and Neslin, S. A. (2008). *Why database marketing?* Springer.

Blum, C. and Roli, A. (Sept. 2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.* **35** (3), pp. 268–308. DOI: 10.1145/937503.937505.

Bolt, A. and Sepúlveda, M. (2014). Process remaining time prediction using query catalogs. In: *Business Process Management Workshops: BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers 11*. Springer, pp. 54–65.

Bose, R. J. C., van der Aalst, W. M., Žliobaitė, I., and Pechenizkiy, M. (2011). Handling concept drift in process mining. In: *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings 23*. Springer, pp. 391–405.

Bose, R. J. C., van der Aalst, W. M., Žliobaitė, I., and Pechenizkiy, M. (2013). Dealing with concept drifts in process mining. *IEEE transactions on neural networks and learning systems* **25** (1), pp. 154–171.

Bozorgi, Z. D., Teinemaa, I., Dumas, M., La Rosa, M., and Polyvyanyy, A. (2021). Prescriptive process monitoring for cost-aware cycle time reduction. In: *2021 3rd International Conference on Process Mining (ICPM)*. IEEE, pp. 96–103.

Bradley, W. and Henseler, J. (2007). Modeling reflective higher-order constructs using three approaches with PLS path modeling: a Monte Carlo comparison.

Breiman, L. (2001a). Random forests. *Machine learning* **45**, pp. 5–32.

Breiman, L. (2001b). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science* **16** (3), pp. 199–231.

Camargo, M., Dumas, M., and González-Rojas, O. (July 2019a). Learning Accurate LSTM Models of Business Processes. In: pp. 286–302. DOI: 10.1007/978-3-030-26619-6_19.

Camargo, M., Dumas, M., and Rojas, O. G. (2019b). Simod: A Tool for Automated Discovery of Business Process Simulation Models. In: *BPM (PhD/Demos)*, pp. 139–143.

Camargo, M., Dumas, M., and González-Rojas, O. (Mar. 2020). Automated discovery of business process simulation models from event logs. *Decision Support Systems* **134**, p. 113284. DOI: 10.1016/j.dss.2020.113284.

Center), A. (P. Q. (2022). *End-to-End Process Maps and Measures (APQC)*. URL: https://www.apqc.org/resource-library/resource-collection/end-end-process-maps-and-measures (visited on 10/18/2022).

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., et al. (2000). CRISP-DM 1.0: Step-by-step data mining guide. *SPSS inc* **9**, p. 13.

Chen, B., Wen, M., Shi, Y., Lin, D., Rajbahadur, G. K., and Jiang, Z. M. (2022). Towards training reproducible deep learning models. In: *Proceedings of the 44th International Conference on Software Engineering*, pp. 2202–2214.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 785–794. DOI: 10.1145/2939672.2939785.

Crooks, A. T. and Heppenstall, A. J. (2011). Introduction to agent-based modelling. In: *Agent-based models of geographical systems*. Springer, pp. 85–105.

Davenport, T. H., Short, J. E., et al. (1990). The new industrial engineering: information technology and business process redesign.

Davenport, T. H. et al. (2006). Competing on analytics. *Harvard business review* **84** (1), p. 98.

Di Francescomarino, C. and Ghidini, C. (2022). Predictive process monitoring. *Process Mining Handbook. LNBIP* **448**, pp. 320–346.

Diao, Y., Jan, E., Li, Y., Rosu, D., and Sailer, A. (2016). Service analytics for IT service management. *IBM Journal of Research and Development* **60** (2-3), 13:1–13:17. DOI: 10.1147/JRD.2016.2520620.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80.

Dumas, M., La Rosa, M., Mendling, J., Reijers, H. A., et al. (2018). *Fundamentals of business process management*. Vol. 2. Springer.

East, R., Romaniuk, J., and Lomax, W. (2011). The NPS and the ACSI: A critique and an alternative metric. *International Journal of Market Research* **53** (3), pp. 327–346.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (Apr. 2004). Least angle regression. *The Annals of Statistics* **32** (2). DOI: 10.1214/009053604000000067.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural Architecture Search: A Survey. *Journal of Machine Learning Research* **20** (55), pp. 1–21. URL: http://jmlr.org/papers/v20/18-598.html.

Evermann, J., Rehse, J. R., and Fettke, P. (2016). A deep learning approach for predicting process behaviour at runtime. *International Conference on Business Process Management* **1**, p. 490. DOI: 10.1007/978-3-319-58457-7.

Feurer, M. and Hutter, F. (2019). Hyperparameter Optimization. In: *Automated Machine Learning: Methods, Systems, Challenges*. Ed. by F. Hutter, L. Kotthoff, and J. Vanschoren. Cham: Springer International Publishing, pp. 3–33. DOI: 10.1007/978-3-030-05318-5_1.

Fishman, G. S. (2001). *Discrete-event simulation: modeling, programming, and analysis*. Berlin: Springer-Verlag. DOI: 10.1017/978-1-4757-3552-9.

Fitz-Gibbon, C. T. (1990). *Performance indicators*. Vol. 2. Multilingual Matters.

Fluxicon BV (Oct. 18, 2022). *Disco*. Version 3.3.7. URL: https://fluxicon.com/disco/.

Folino, F., Guarascio, M., and Pontieri, L. (2013). Discovering High-Level Performance Models for Ticket Resolution Processes: (Short Paper). In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences: Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*. Springer, pp. 275–282.

Francescomarino, C. D., Dumas, M., Federici, M., Ghidini, C., Maggi, F. M., Rizzi, W., and Simonetto, L. (2018). Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems* **74**. Information Systems Engineering: selected papers from CAiSE 2016, pp. 67–83. DOI: https://doi.org/10.1016/j.is.2018.01.003.

Galup, S. D., Dattero, R., Quan, J. J., and Conger, S. (2009). An overview of IT service management. *Communications of the ACM* **52** (5), pp. 124–127.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2013). A Survey on Concept Drift Adaptation. *ACM Computing Surveys*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Hallowell, R. (1996). The relationships of customer satisfaction, customer loyalty, and profitability: an empirical study. *International journal of service industry management*.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Hinkka, M., Lehto, T., and Heljanko, K. (2020). Exploiting event log event attributes in RNN based prediction. In: *Data-Driven Process Discovery and Analysis: 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13–14, 2018, and 9th International Symposium, SIMPDA 2019, Bled, Slovenia, September 8, 2019, Revised Selected Papers 8*. Springer, pp. 67–85.

Hlupic, V. and Robinson, S. (1998). Business process modelling and analysis using discrete-event simulation. In: *1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274)*. Vol. 2. IEEE, pp. 1363–1369.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation* **9** (8), pp. 1735–1780.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2013). An Evaluation of Sequential Model-Based Optimization for Expensive Blackbox Functions. In: *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*. GECCO '13 Companion. Amsterdam, The Netherlands: Association for Computing Machinery, pp. 1209–1216. DOI: 10.1145/2464576.2501592.

Jansen-Vullers, M., Loosschilder, M., Kleingeld, P., and Reijers, H. (2007). Performance measures to evaluate the impact of best practices. In: *Proceedings of Workshops and Doctoral Consortium of the 19th International Conference on Advanced Information*

Systems Engineering (BPMDS workshop). Vol. 1. Tapir Academic Press Trondheim, pp. 359–368.

Karimi, Z., Abolhassani, H., and Beigy, H. (2012). A new method of mining data streams using harmony search. *Journal of Intelligent Information Systems* **39**, pp. 491–511.

Keiningham, T. L., Cooil, B., Andreassen, T. W., and Aksoy, L. (2007). A Longitudinal Examination of Net Promoter and Firm Revenue Growth. *Journal of Marketing* **71** (3), pp. 39–51. DOI: 10.1509/jmkg.71.3.039.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University* **33** (2004), pp. 1–26.

Krumeich, J., Werth, D., and Loos, P. (2016). Prescriptive control of business processes. *Business & Information Systems Engineering* **58** (4), pp. 261–280.

Kubrak, K., Milani, F., Nolte, A., and Dumas, M. (2022). Prescriptive process monitoring: Quo vadis? *PeerJ Computer Science* **8**, e1097.

Kumar, P. and Batra, S. (Oct. 2018). Meta-heuristic based Optimized Deep Neural Network for Streaming Data Prediction. In: DOI: 10.1109/ICACCCN.2018.8748691.

Kumar, V. and Shah, D. (2004). Building and sustaining profitable customer loyalty for the 21st century. *Journal of retailing* **80** (4), pp. 317–329.

La Rosa, M. and Soffer, P. (2013). *Business Process Management Workshops: BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012, Revised Papers*. Vol. 132. Springer.

Lepenioti, K., Bousdekis, A., Apostolou, D., and Mentzas, G. (2020). Prescriptive analytics: Literature review and research challenges. *International Journal of Information Management* **50**, pp. 57–70. DOI: https://doi.org/10.1016/j.ijinfomgt.2019.04.003.

Levinson, W. (2010). *Statistical Process Control for Real-World Applications*. Taylor & Francis.

López-Pintado, O. and Dumas, M. (2022). Business Process Simulation with Differentiated Resources: Does it Make a Difference? In: *Business Process Management: 20th International Conference, BPM 2022, Münster, Germany, September 11–16, 2022, Proceedings*. Springer, pp. 361–378.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems* **30**.

Maclaurin, D., Duvenaud, D., and Adams, R. P. (2015). Gradient-Based Hyperparameter Optimization through Reversible Learning. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, pp. 2113–2122.

Maisenbacher, M. and Weidlich, M. (June 2017). Handling Concept Drift in Predictive Process Monitoring. In: pp. 1–8. DOI: 10.1109/SCC.2017.10.

Mannhardt, F., de Leoni, M., Reijers, H., and Aalst, W. (Feb. 2015). Balanced multi-perspective checking of process conformance. *Computing*. DOI: 10.1007/s00607-015-0441-1.

Mannhardt, F. and Blinde, D. (2017). Analyzing the Trajectories of Patients with Sepsis using Process Mining. *RADAR+ EMISA@ CAiSE* **1859**, pp. 72–80.

Márquez-Chamorro, A. E., Nepomuceno-Chamorro, I. A., Resinas, M., and Ruiz-Cortés, A. (2022). Updating prediction models for predictive process monitoring. In: *Advanced Information Systems Engineering: 34th International Conference, CAiSE 2022, Leuven, Belgium, June 6–10, 2022, Proceedings*. Springer, pp. 304–318.

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: http://tensorflow.org/.

McCormack, K. P. (1999). *The development of a measure of business process orientation and its link to the interdepartmental dynamics construct of market orientation*. Nova Southeastern University.

Mesabbah, M. and McKeever, S. (2018). Presenting a hybrid processing mining framework for automated simulation model generation. In: *2018 Winter Simulation Conference (WSC)*. IEEE, pp. 1370–1381.

Muñoz, M. A., Sun, Y., Kirley, M., and Halgamuge, S. K. (2015). Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences* **317**, pp. 224–245. DOI: https://doi.org/10.1016/j.ins.2015.05.010.

Navarin, N., Vincenzi, B., Polato, M., and Sperduti, A. (2018). LSTM networks for data-aware remaining time prediction of business process instances. *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings* **2018-Janua**, pp. 1–7. DOI: 10.1109/SSCI.2017.8285184.

Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa (2017). Predictive Business Process Monitoring with LSTM Neural Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **10253 LNCS**, pp. V–VI. DOI: 10.1007/978-3-319-59536-8.

Nisafani, A., Wibisono, A., Kim, S., and Bae, H. (Feb. 2012). Bayesian Selection Rule for Human-Resource Selection in Business Process Management Systems. *The Journal of Society for e-Business Studies* **17**, pp. 53–74. DOI: 10.7838/jsebs.2012.17.1.053.

O'Neill, P. and Sohal, A. S. (1999). Business Process Reengineering A review of recent literature. *Technovation* **19** (9), pp. 571–581.

Omar, H. K., Jihad, K. H., and Hussein, S. F. (2021). Comparative analysis of the essential CPU scheduling algorithms. *Bulletin of Electrical Engineering and Informatics* **10** (5), pp. 2742–2750.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika* **41** (1/2), pp. 100–115.

Pepper, M. P. and Spedding, T. A. (2010). The evolution of lean Six Sigma. *International Journal of Quality & Reliability Management*.

Porter, M. E. (1985). *Competitive advantage: creating and sustaining superior performance*. eng. New York: London: Free Press; Collier Macmillan.

Railsback, S. F., Lytinen, S. L., and Jackson, S. K. (2006). Agent-based simulation platforms: Review and development recommendations. *Simulation* **82** (9), pp. 609–623.

Rama-Maneiro, E., Vidal, J., and Lama, M. (2020). Deep Learning for Predictive Business Process Monitoring: Review and Benchmark. *ArXiv* **abs/2009.13251**.

Reichheld, F. F. (2003). The one number you need to grow. *Harvard Business Review* **81** (12), pp. 46–55.

Rubinstein, R. Y. and Kroese, D. P. (2016). *Simulation and the Monte Carlo Method*. 3rd. Wiley Publishing.

Sheth, J. N. and Parvatiyar, A. (1995). The evolution of relationship marketing. *International business review* **4** (4), pp. 397–418.

Shoush, M. and Dumas, M. (2022). Prescriptive process monitoring under resource constraints: a causal inference approach. In: *Process Mining Workshops: ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31–November 4, 2021, Revised Selected Papers*. Springer, pp. 180–193.

Slack, N., Brandon-Jones, A., and Johnston, R. (June 2016). *Operations Management, 8th edition*. English. 8th. Pearson.

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of business research* **104**, pp. 333–339.

Szimanski, F., Ralha, C. G., Wagner, G., and Ferreira, D. R. (2013). Improving business process models with agent-based simulation and process mining. In: *Enterprise, Business-Process and Information Systems Modeling: 14th International Conference, BPMDS 2013, 18th International Conference, EMMSAD 2013, Held at CAiSE 2013, Valencia, Spain, June 17-18, 2013. Proceedings*. Springer, pp. 124–138.

Tan, K. W., Wang, C., and Lau, H. C. (2012). Improving patient flow in emergency department through dynamic priority queue. In: *2012 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, pp. 125–130.

Teinemaa, I., Tax, N., Leoni, M. d., Dumas, M., and Maggi, F. M. (2018a). Alarm-based prescriptive process monitoring. In: *International Conference on Business Process Management*. Springer, pp. 91–107.

Teinemaa, I., Dumas, M., Leontjeva, A., and Maggi, F. M. (2018b). Temporal stability in predictive process monitoring. *Data Mining and Knowledge Discovery* **32** (5), pp. 1306–1338. DOI: 10.1007/s10618-018-0575-9.

Teniente, E. and Weidlich, M. (2018). *Business Process Management Workshops: BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*. Vol. 308. Springer.

Thomas, L., Kumar, M. V. M., and Annappa, B. (2017). Recommending an Alternative Path of Execution Using an Online Decision Support System. In: *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics Swarm Intelligence*. ISMSI '17. Hong Kong, Hong Kong: Association for Computing Machinery, pp. 108–112. DOI: 10.1145/3059336.3059361.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* **58** (1), pp. 267–288. URL: http://www.jstor.org/stable/2346178.

TMForum (2023). *End-to-End Process Maps and Measures (APQC)*. URL: https://www.tmforum.org/oda/business/process-framework-etom/ (visited on 05/03/2023).

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Technical Report TCD-CS-2004-15, Trinity College Dublin, 58*.

Tutz, G., Schmid, M., et al. (2016). *Modeling discrete time-to-event data*. Springer.

van der Aalst, W., Weijters, T., and Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE transactions on knowledge and data engineering* **16** (9), pp. 1128–1142.

van der Aalst, W. (Apr. 2015). Business Process Simulation Survival Guide. *Handbook on Business Process Management 1. International Handbooks on Information Systems.*, pp. 337–370. DOI: 10.1007/978-3-642-45100-3_15.

van der Aalst, W. M. P. (2016). *Process Mining: Data Science in Action*. 2nd ed. Heidelberg: Springer. DOI: 10.1007/978-3-662-49851-4.

van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H., Weijters, A., and van Der Aalst, W. M. (2005). The ProM framework: A new era in process mining tool support. In: *Applications and Theory of Petri Nets 2005: 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005. Proceedings 26*. Springer, pp. 444–454.

van Dongen, B. F., Crooy, R. A., and van der Aalst, W. M. (2008). Cycle time prediction: When will this case finally be finished? In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, pp. 319–336.

Verenich, I. (Dec. 2016). Helpdesk. *Mendeley data*. DOI: 10.17632/39bp3vv62t.1.

Verenich, I., Dumas, M., Rosa, M. L., Maggi, F. M., and Teinemaa, I. (2019). Survey and Cross-benchmark Comparison of Remaining Time Prediction Methods in Business Process Monitoring. *ACM Transactions on Intelligent Systems and Technology* **10** (4), pp. 1–34. DOI: 10.1145/3331449.

Wang, E., Li, D., Dong, B., Zhou, H., and Zhu, M. (2020). Flat and hierarchical system deployment for edge computing systems. *Future Generation Computer Systems* **105**, pp. 308–317.

Wibisono, A., Nisafani, A. S., Bae, H., and Park, Y.-J. (2015). On-the-fly performance-aware human resource allocation in the business process management systems environment using naıve bayes. In: *Asia Pacific Business Process Management: Third Asia Pacific Conference, AP-BPM 2015, Busan, South Korea, June 24-26, 2015, Proceedings 3*. Springer, pp. 70–80.

Žliobaitė, I., Pechenizkiy, M., and Gama, J. (2016). An Overview of Concept Drift Applications. In:

Zou, H. and Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **67** (2), pp. 301–320. URL: http://www.jstor.org/stable/3647580.

Appended Papers

Paper I

Riess, M. (2023c). Remaining cycle time prediction: Temporal loss functions and prediction consistency. Manuscript submitted to *Nordic Machine Intelligence*.

NORDIC MACHINE INTELLIGENCE

Nordic Machine Intelligence, vol. ??, pp. 1–16, 2023
Received 15 Feb 2023 / published dd mmm yyyy
https://doi.org/10.2478/nmi-some-doi

Remaining cycle time prediction: Temporal loss functions and prediction consistency

Mike Riess [1]

1. *E-mail any correspondence to: mike.riess@nmbu.no*

Abstract
The usefulness of remaining cycle time models for predictive and prescriptive process monitoring depends not only on the overall accuracy of the predictions but also on their earliness: predictions should be as accurate as possible, as early as possible. To give this criterion more weight in model fitting, the paper evaluates three L1 loss functions with temporal decay. All have the property of increasing the weight of residuals from the early stages of a process relative to residuals from later stages but do so to different degrees. The loss functions are used in LSTM networks for training remaining throughout time models of four different business processes based on publicly available event log data sets. Compared to models trained with unweighted L1 loss, the suggested modifications yield small but significant improvements in earliness on out-of-sample data. Neither the unweighted L1 loss nor the modifications led to models with strictly monotonically decreasing predictions of the remaining time.

Keywords: Predictive process monitoring, LSTM, temporal loss function, temporal consistency, remaining time prediction

Introduction

In predictive process monitoring [1], the main goal is to generate predictions that can support process interventions before an undesired event is likely to occur. These interventions can either be manually initiated or automated via *prescriptive* process monitoring [2]. Possible prediction targets include the expected process outcome [3], the next event type [4], the current event duration [4], the total remaining cycle time [4, 5], and other useful process monitoring information. When using remaining cycle time for decision support in areas such as queue prioritization [6, 7, 8] or dynamic resource planning [9, 10, 11, 12], the accuracy of a prediction is most important at early stages of a process when there is little information available to base it on. This is due to the sequential nature of business processes [13], where process-related information accumulates over time as activities are performed. At the same time, the ideal resource plan is based on accurate information that is available before the plan is needed or executed.

Remaining cycle time prediction is a well-studied problem in the literature on predictive process monitoring. The main goal here is to predict the remaining time before an ongoing process instance (for example a customer service case) is completed. Process data usually have the structure of time-stamped discrete-event logs with varying time between events [1] and can therefore not be modelled as conventional time series data. Multiple approaches have been proposed to solve this problem, reviewed in [1] and [14]. These comparisons show that the prefix log data format (a "long" data format where all time steps preceding a focal event are represented as observations in the data set; for details, see below) combined with long short-term memory recurrent neural networks (LSTM-RNN) [5] yields the on average highest overall accuracy across the evaluated application areas.

Organizations using remaining time models for case prioritization or dynamic planning need accurate predictions in the beginning of a case, when information about the case itself is minimal, and prediction is the hardest. This performance aspect of a prediction model is referred to as *earliness* [1] in the literature.

The LSTM approaches reviewed in [4, 5] as well as the time-based process prediction models evaluated in [14] are all trained using the L1 loss, also known as the mean absolute error (MAE) loss function. The advantage of this loss function is that it is not sensitive to large variation in the time between events, compared to other loss functions such as the mean squared error (MSE) [1]. In the literature on remaining time prediction, it is most common to evaluate predictions using the MAE [4, 5, 14]. Using the MAE also as the loss function

will theoretically lead to the highest overall accuracy (weighing all events equally). In their experiments, the authors of [15] find a trade-off between accuracy and earliness, which they address using an ensemble approach (combining multiple models to make a single prediction, also referred to as model averaging in the literature). Another approach using genetic algorithms for automated hyper-parameter optimization was proposed by [16]. The authors incorporate earliness as one term in the fitness function used for hyper-parameter optimization. However, altering the loss function used for training the model itself, and the effect on earliness of such a strategy, has to the best knowledge of the author not yet been studied. For the LSTM-based models using the approach suggested by [5], the effect of temporal weighting of the loss function on the earliness performance of the resulting prediction models is therefore still unknown.

Another aspect of the approach in [5] is that the formulation of the prediction problem (target) and the specification of the loss function do not require that consecutive predictions represent the "natural behaviour" of time. As time passes, the true remaining time will monotonically decrease. However, the true remaining time is not known before a case has been completed. As more information about an ongoing case becomes available over time, the predicted remaining time may increase if the new information indicates this. Although it is currently unknown how much LSTM-based remaining time models respond to this, the fact that they do not restrict the functional form of the relationship between input features and targets suggests that they should respond more strongly than more restrictive models. Somewhat related, the temporal stability of analogous binary classification models has been studied by [17]. However, this approach does not involve tests of the expected monotonic decrease of remaining time predictions. In practical terms, this is important because temporal inconsistencies may lead to ambiguous decision support, as the direction of the predicted remaining time for a case might change. If used in the context of dynamic work shift planning, unreliable estimates could lead to irregular or extended work shifts, which is known to have negative consequences for the workforce [18, 10, 19, 11] and may increase operating costs [20].

Research questions

Th research reported in this paper will address two questions:

- **RQ1:** How do temporally weighted loss functions influence the performance of LSTM-based remaining time prediction models?
- **RQ2:** To which degree do the predictions generated by LSTM-based remaining time prediction models fulfill the criterion of temporal consistency?

To answer *RQ1* the currently best-performing approach to remaining time prediction [5] is utilized using three modified versions of the mean absolute error loss. The proposed loss functions use temporal decay to induce different degrees of the relative importance of early residuals. *RQ2* is answered via the introduction of an additional evaluation aspect, *temporal consistency (TC)*. The proposed metric will be used to evaluate the trained models in terms of their ability to generate monotonically decreasing predictions of remaining cycle time.

Key concepts

Process mining and predictive process monitoring

Predictive process monitoring is *"a multi-disciplinary area that draws concepts from process mining on one side, and machine learning on the other"* [1]. Process mining [21] is a sub-field of business process management (BPM) [22] and is concerned with the analysis of processes based on event data extracted from the management information systems of an organization. The main types of applications include process discovery, conformance checking, process re-engineering, and operational support [23]. Predictive process monitoring [3] mainly relates to operational support. Here, the main goal is to use process data to train machine learning models [24] to predict currently unknown characteristics about the outcome of a process (duration, activities, conformance, etc.) before the outcome is realized. These predictions are thereby intended to help the organization make *proactive* rather than *reactive* managerial decisions [21].

Event log data

Event log data consist of time-stamped pieces of information related to distinct *cases* or *instances* of a business process. Event log data is most often found in process-aware information systems [21] such as enterprise resource planning (ERP) and customer relationship management (CRM) systems. Table 1 shows part of an example event log similar to what can be found in the case management modules of common CRM systems.

Definition (Event): An event e is a tuple $(a, c, t, (d_1, v_1), ..., (d_m, v_m))$ where a is an activity, c is a case identifier, t is an associated timestamp, and d_m is a set of attributes with their associated values v_m. An event can have multiple timestamps, for example at the start and end of the event. In this case, the tuple would be extended to have the form: $(a, c, t_{start}, t_{end}, (d_1, v_1), ..., (d_m, v_m))$.

The sequence of events generated by a given process instance or case forms a *trace*. A trace contains events related to a single case and consist of a case identifier (unique), event identifiers, timestamps, resources, and other associated attributes.

Definition (Trace): A trace is a non-empty sequence $Q = \langle e_1, ..., e_T \rangle$ of events from a *case* such that $\forall t \in [1, ..., T]$, $e_t \in \varepsilon$, where ε is the universe of all possible

Case ID	Case topic	Activity	Timestamp	Resource
1001	Invoice	Email interaction	01-01-2019 15:01	System
1001	Invoice	Phone interaction	02-01-2019 16:04	User 2
1001	Invoice	Change data	04-01-2019 16:58	User 1
1002	Service malfunction	Email interaction	01-01-2019 12:01	System
1002	Service malfunction	Phone interaction	03-01-2019 13:10	User 2
1002	Service malfunction	Change data	03-01-2019 14:15	User 5
1002	Service malfunction	Send invoice	04-01-2019 09:35	User 2

Table 1: Example event-log in a hypothetical customer service unit.

events. All events in a trace refer to the same case identifier c.

A set of traces constitutes an event log.

Definition (Event log): An event log L is a set of K traces of length T_i: $L = \{Q_i : Q_i \in S, 1 \leq i \leq K\}$, where i is the trace enumerator, S the universe of all possible traces, and K the number of traces in the event log.

Prefix logs

In order to represent the state of a trace at the time of each recorded event, the prefix of a given trace is defined as a set of partially observed subsets of the completed trace [1]. This enables machine learning models to learn patterns from the development of the traces over time.

Definition (Prefix function): Given a trace of length T, $Q = \{e_1, ..., e_T\}$, a positive integer $k \leq T$, and the *head* operator hd^k, the prefix function selects the first k events in the trace up until event k: $hd^k(Q) = \langle e_1, ..., e_k \rangle$.

For every trace of length K, a prefix function will thus map a trace onto K different prefixes, where each prefix $hd^k(Q)$ contains the full trace up until the kth event. Performing this procedure for every trace in an event log transforms it into a prefix log.

Definition (Prefix log): Given an event log L, its prefix log L^* is the event log that contains all prefixes of L: $L^* = \{hd^k(Q) : Q \in L, 1 \leq k \leq \|Q\|\}$.

The *prefix log* thus contains a set of partially observed traces (prefixes), which represent the "history" of a trace for the period in which it has been active, up until the focal event.

Remaining time prediction

A central task in the field of predictive process monitoring is the prediction of the remaining cycle time of an open case or ongoing process instance [1]. This is usually achieved using a machine learning model trained on a prefix log X, where each prefix i is treated as an observation. The target value y at the tth event in a trace is defined as the remaining time until the last event in the trace, as shown in Equation 1. In this case, e_t denotes the current event t, and e_T denotes the final event in the trace.

$$y_t = e_T(t_{start}) - e_t(t_{start}) \tag{1}$$

As some event logs only have timestamps for event start time, the definition in equation 1 is the most commonly used in the literature [4, 5, 1].

Long short-term memory recurrent neural networks

A long short-term memory recurrent neural network (LSTM-RNN) [25] is a recurrent neural network (RNN) with modified hidden units, sometimes referred to as LSTM cells or blocks. This network architecture is designed to learn long patterns by adaptively changing the amount of information that is remembered or forgotten via input, output, and forget gates. An LSTM cell has five components: input gate i_t, forget gate f_t, cell state c_t, output gate o_t, and the final output of the cell itself h_t. The elements W_{xi}, W_{hi}, W_{ci}, W_{xf}, W_{hf}, W_{cf}, W_{xc}, W_{hc}, W_{x0}, W_{h0}, W_{c0} all denote weight matrices, and b_i, b_f, b_c, b_0 denote the biases of the four weighted components (equations 2 to 5).

$$i_t = sigmoid(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2}$$

$$f_t = sigmoid(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{3}$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc} + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$o_t = sigmoid(W_{x0}x_t + W_{h0}h_{t-1} + W_{c0}c_t + b_0) \tag{5}$$

$$h_t = o_t \tanh(c_t) \tag{6}$$

The gates in the cell architecture are designed to avoid the vanishing/exploding gradient problem often encountered in the process of training RNNs, as described in [26]. An LSTM is trained using backpropagation through time (BPTT) [25] and stochastic gradient descent. As shown in Equation 2, the input gate i_t receives the information from the input data X_t, the previous hidden state h_{t-1}, and the previous cell state c_{t-1}. This enables each cell to learn from the information contained in previous steps (controlled by the forget gate f_t and the output gate o_t), and thereby gives it a long short-term memory. This is specifically beneficial in problems where the beginning of a long sequence carries information that is relevant for predicting the end of the sequence [4].

Architecture

Figure 1 shows a recurrent LSTM neural network with two layers, at the time the model makes its prediction. The network has a single output node, which is the linear combination of the output gates of h_{t+1}, where t is the event enumerator (going from left to right). In this example, the network has a sequence length of four events, and the illustration shows the inputs at event t. In the data X_{t+1} representing the next step (which has not yet been observed), the missing values are replaced by zero ("zero-padded"; see [1]).

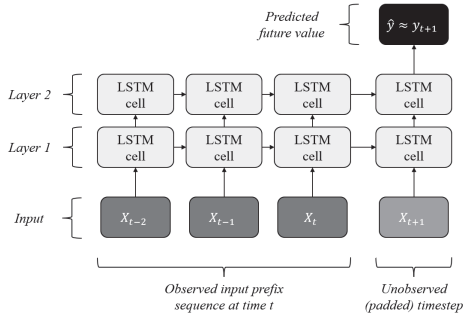


Figure 1: LSTM neural network architecture with a single output and two LSTM layers.

Loss functions

The most commonly used loss function for remaining time prediction in predictive process monitoring [4, 5, 1] is the mean absolute error (MAE). This loss is defined as the $L1$ norm of the difference between the prediction y_t^i at event t in trace i, and the actual target value $\hat{y}_t^i$:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T_i} \mid y_t^i - \hat{y}_t^i \mid \qquad (7)$$

The MAE is more robust against outliers than the mean squared error (where the difference between target and predicted value is squared). The standard formulation of the MAE loss is time-invariant but has performed best in the analyses of event log data with large time differences between events [1].

Temporally weighted loss functions

The purpose of the temporally weighted loss functions is to force the model training process to incorporate earliness as a performance criterion (in addition to overall accuracy, assuming there is a trade-off between the two [1, 15]). Since a temporal decay factor is introduced into the loss functions, minimizing early errors in a sequence of prefixes will reduce total loss relatively more than minimizing later errors.

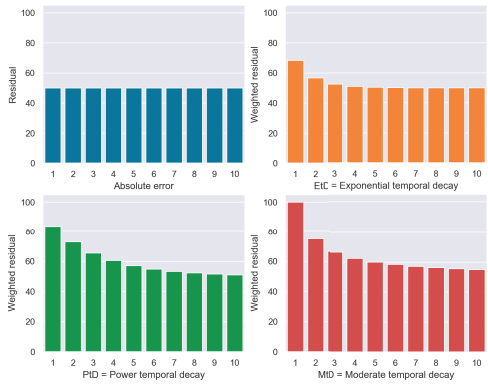


Figure 2: Temporal decay of the temporally weighted loss functions given a constant error of 50 at each event in a trace of length ten.

To assess the usefulness of temporal weighting for improving the *earliness* properties of remaining time models, three modified variants of the mean absolute error are proposed in the following. The variants differ in the degree of temporal decay imposed on the mean absolute error. This is achieved using different functional forms. Figure 2 illustrates the effect of the three temporal weighting schemes, compared to the unweighted mean absolute error. The illustration shows how the loss functions would transform a constant error of 50 at multiple prefixes. In this particular example, the temporal decay makes up 5.8, 20 and 29 percent respectively of the sum of the residuals at all ten prefixes. However, the proportions will vary depending on the size and inter-temporal variation of the residuals.

The mean absolute error with *exponential temporal decay*, MAE_{EtD} (depicted in orange in Fig. 2), is defined as the mean absolute error with a temporal penalty term added in the form of an exponential decay of the absolute errors, as seen in Equation 8.

$$MAE_{EtD} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T_i} \mid y_t^i - \hat{y}_t^i \mid + \frac{\mid y_t^i - \hat{y}_t^i \mid}{e^{(t)}} \qquad (8)$$

As t (the time step or prefix enumerator) gets larger, the weight of the residual in this case decreases exponentially by the factor $e^{(t)}$. Compared to the MAE_{MtD} and MAE_{PtD}, this loss has the lowest peak and the fastest decay, getting close to the standard MAE after approximately three events, as seen in Fig. 2.

The mean absolute error with *power-based temporal decay* (MAE_{PtD}) uses the power of the remaining number of prefixes in trace i ($\frac{T_i - t}{T_i}$) to weigh the residuals at each prefix t, as shown by Equation 9.

$$MAE_{PtD} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T_i} \mid y_t^i - \hat{y}_t^i \mid + \mid y_t^i - \hat{y}_t^i \mid^{\frac{T_i - t}{T_i}} \quad (9)$$

The mean absolute error with *moderate temporal decay* (MAE_{MtD}) uses the current prefix enumerator t as a weighting factor for the residuals. This leads to relatively higher residuals than MAE_{EtD} and MAE_{PtD} at all prefixes. Of the three proposed loss functions, this is the loss with the largest temporal penalty. The functional form of the MAE_{MtD} is shown in Equation 10.

$$MAE_{MtD} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T_i} \mid y_t^i - \hat{y}_t^i \mid + \frac{\mid y_t^i - \hat{y}_t^i \mid}{t} \quad (10)$$

Materials and methods

In the following sections, the methodology of this study is described in detail. To evaluate the relative performance of the loss functions, a series of experiments were conducted using a full factorial experimental design with two factors (loss function, event log data set; each with four levels). Ten replications were performed per cell of the design. The first factor represents the four different event log data sets, which will be described in more detail in the next sections. The factor levels are listed below.

- **Factor one: Event log data set**
  - Sepsis
  - Helpdesk
  - Traffic fines
  - Hospital billing

The second factor is the loss function, where MAE is seen as the baseline. Three additional loss functions will be evaluated alongside the MAE, and are introduced in the next section.

- **Factor two: Loss function**
  - MAE
  - MAE_{EtD}
  - MAE_{PtD}
  - MAE_{MtD}

Data

Four event log data sets from different domains are used in this study to evaluate the performance of the temporal loss functions. All data sets are publicly available and have been used as "benchmark data" in several previous studies in the process mining community [27, 5, 28, 29]. These four particular event logs were selected to illustrate different sectors/areas, as well as different process characteristics with varying degrees of complexity and traces. Qualitative descriptions of the individual event logs can be seen in Table 2, while descriptive statistics can be found in Table 3.

As seen by figure 3, the distribution of the *Sepsis* data stands out due to the low number of traces, as well as the distribution of the prefix lengths. Compared to the rest of the event-logs, the *Sepsis* data has a longer tail, and thereby a higher frequency of relatively long traces. As described in the next section, the event-logs have been truncated to length 20, such that they maximally include the first 20 events of a trace. This unfortunately reduces the external validity of the results in the *Sepsis* and *Traffic fines* data, as well as the difference in the trace length distributions across the four event-logs. The motivation for truncation was to reduce computational load as in [17], and avoid statistical uncertainty for long traces with low support as in [1]. However, looking at the average trace lengths in table 3, the majority of the traces preserve their original trace length using truncation after 20 events.

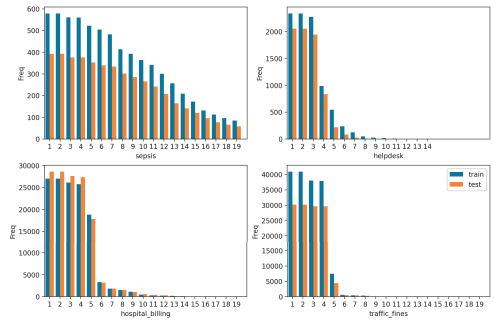


Figure 3: Distribution of prefixes across pre-processed event-log data (the last event of each trace is excluded as described in table 4).

Data preparation

Prior to model training, the event-log data have been prepared using a set of operations listed in table 4 below. All event-logs have been processed in the same way, and share the same format and feature types.

Hyper-parameters

To ensure that the optimal hyper-parameters were used in each of the four event-logs, a set of grid-search experiments were performed within the training period (see fig 4) prior to the main experiments. The grid-search was performed using the *MAE* loss function, to ensure ideal conditions for the baseline loss function. The search space included the learning rate (0.01, 0.1, 0.15, 0.2), number of LSTM-layers (1, 2, 3), optimization algorithm (Stochastic gradient descent [24] and Adaptive moment estimation with Nesterov momentum [33]) and finally batch size (128, 512, 1024, 2048). Settings similar to these have been used in previous studies [1, 5, 4]. Following the implementation in [5], each LSTM-layer

Event log	Area	Description
Sepsis	Healthcare	Event log containing events of sepsis cases (life-threatening condition typically caused by an infection) from a municipality hospital [28].
Helpdesk	IT Services	Case data from the ticket management system of an Italian software company [30, 5].
Traffic fines	Public administration	Event log from an information system managing road traffic fines [29].
Hospital billing	Healthcare/Financial accounting	Event log data obtained from the financial modules of the Enterprise Resource Planning (ERP) system in a regional hospital [27].

Table 2: Qualitative overview of the event-log data.

Event-log	n cases	n censored	Max trace length	Avg. trace length	Avg. case duration	Truncation
Sepsis	1050	78	185	14.49	28.06	20
Helpdesk	4580	180	15	4.66	40.54	20
Traffic fines	150370	79179	20	3.73	341.67	20
Hospital billing	100000	44340	217	4.51	126.99	20

Table 3: Descriptive statistics (full event-log with no partitioning). Durations are measured in days.

was given a recurrent dropout with a *dropout-probability* of 20%. Furthermore, the output of each LSTM-layer was batch-normalized before it was used as input in the next layer, except from the final output $\hat{y}$ (the linear combination of the last cell output o_t in the last layer - see fig 1). The final hyper-parameters for each of the four event-logs can be found in table 5. The source code for performing the grid-search and main experiments can be found at https://github.com/Mikeriess/Temporal_loss.

Evaluation

To evaluate the performance of the trained models, the event-log data have been partitioned into train and test periods using a temporal split as depicted in figure 4. In the example, case one and two are in the train set, case three overlap both periods and is thus censored, while case four is in the test set.The date that separate the two subsets, is the date that split the first event of 50% of the first cases into in the train period, and the remaining 50% is then denoted as the test period. Cases that finish outside either of the two periods are censored (dropped if they do not finish within their starting period). Unfortunately, this leads to a significant drop in the number of cases for the *Hospital billing* (44%) and *Traffic fines* (52%) data (see column *'n censored'* in table 3). However, as the goal of this study is to evaluate out-of-distribution forecasts, it is seen as more important that the train and test distributions are fully separated, than that the original samples are used. Temporal splitting is also used in [1] and [17], however neither of these studies use censoring of overlapping cases.

The performance of the loss functions will be evaluated from three perspectives: Overall accuracy, the ability to make accurate predictions when little information is available (earliness), and the ability to represent time in a natural manner (temporal consistency). These aspects represent different types of prediction quality, but will not be sufficient alone.

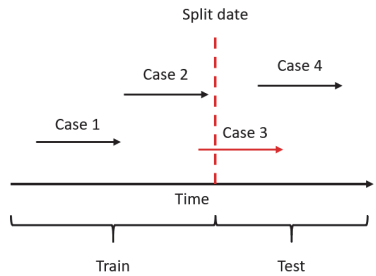


Figure 4: Illustration of the data partitioning scheme.

Accuracy

To evaluate the accuracy of the models, the most commonly used metric in literature is the mean absolute error over all traces in the test set [1], which can be seen in equation 11 below. Here, T_i refers to the number of events in the *i'th* trace, and N to the number of traces in the event-log. The predicted remaining time of the *t'th* event of trace i is denoted by $\hat{y}_t^i$, and it's ground truth as y_t^i.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T_i} \mid y_t^i - \hat{y}_t^i \mid \qquad (11)$$

The accuracy shows the average performance across all traces but does not illustrate potential temporal differences in performance (across prefixes), or segments of the data where the model performance might vary.

Earliness

Evaluating the performance across different points in time, where different degrees of information will be available, the model *earliness* is evaluated by calculating the MAE_t (equation 12). Every *t'th* prefix is thereby measured independently of all prior or subsequent prefixes in the test period.

$$MAE_t = \frac{1}{N}\sum_{i=1}^{N} \mid y_t^i - \hat{y}_t^i \mid \qquad (12)$$

Here, t is the event number in each trace, and i denotes the *i'th* trace in the test period. This approach have been used in related literature on both remaining time prediction, as well as classification problems with data in event-log format [4, 1, 32, 17]. In addition, the MAE_t is also calculated at different trace lengths to investigate potential performance differences in this aspect. In future work, differences across trace variants and case types might also be further studied.

Temporal consistency

To understand the degree to which the prediction models represent the *natural behavior of time*, a new evaluation metric defined as the Temporal Consistency (TC) is proposed in the following. For two consecutive remaining cycle time (ground truth) values $y_{t=0}$ and $y_{t=1}$ observed in an event-log, it always holds that $y_{t=0} \geq y_{t=1}$, as remaining time monotonically decreases as $t \to \infty$. The Temporal Consistency of the remaining cycle time is therefore defined as the absolute first difference $\mid \hat{y}_t - \hat{y}_{t-1} \mid$, when the first difference between two predictions is positive $(\hat{y}_t - \hat{y}_{t-1}) > 0$.

The objective of this metric is to measure the degree of *consistency* in consecutive predictions of a given trace by measuring the predicted values $\hat{y}_t$ alone. From the temporal consistency aspect, ideal model behavior is therefore monotonically decreasing remaining times for each new event in the same trace. As illustrated by equation 13, the TC does neither measure model accuracy nor earliness and is thereby merely proposed as an additional perspective on model performance.

$$TC = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T-1}\sum_{t=2}^{T_i} H\left(\hat{y}_t^i - \hat{y}_{t-1}^i\right) \mid \hat{y}_t^i - \hat{y}_{t-1}^i \mid \qquad (13)$$

where:

$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The aggregate measure in equation 13 can be interpreted as the *average increase in predicted remaining time across all traces*. However, to understand these errors at different prefixes, the TC_t (for all $t > 1$) can also be calculated as shown by equation 14.

$$TC_t = \frac{1}{N}\sum_{i=1}^{N} H\left(\hat{y}_t^i - \hat{y}_{t-1}^i\right) \mid \hat{y}_t^i - \hat{y}_{t-1}^i \mid \qquad (14)$$

Results

In the following sections, the results will be presented with respect to each of the three performance aspects: Accuracy, which will be presented in the next section, Earliness, in the following, and lastly Temporal consistency.

Accuracy performance

Looking at the average error across all traces in the test period, the results seem to vary across the event-logs. In general, the temporal loss functions have the best performance across all four event-logs, however, the magnitude of improvement over baseline (the MAE loss function) varies across event-logs.

From table 6, for the Sepsis data, it can be seen that even though the results show that MAE_{EtD} has the best average performance, the difference from the baseline is very low. For the Helpdesk data, the results are similar, but for the Traffic fines data there is an average performance improvement of 5.22 days. For the Hospital billing data, the difference is again low or non-existing. From the F-test results in table 7, it can be observed that only MAE_{EtD} has significantly different results.

Earliness performance

Compared to the average accuracy, the difference between the loss functions is generally larger from the earliness perspective. Looking at the results in table 8, the performance of the temporal loss functions is on average better than the baseline MAE-loss, with the MAE_{EtD} outperforming most often across event-logs. For the Sepsis data, the MAE_{EtD} outperforms the baseline at each of the first five prefixes reported. This difference is significant for each of the prefixes except the first. However, the magnitude of the difference is less than a day for each of the prefixes in table 8. Looking at the box plot in fig. 5, this pattern seems to continue for all of the prefixes in the Sepsis data. Moving on to the Helpdesk data, the temporal loss functions are again better than the baseline, however, by less than a day in the first three prefixes. The differences were also not found to be statistically significant, except for the MAE_{MtD} at prefix five. Furthermore, it is seen from figure 5 that the performance of MAE_{EtD} improves as the prefix gets larger, compared to MAE.

For the Traffic fines data, the improvement over baseline is the largest with MAE_{EtD}, with 5.35 days lower error at the first prefix, and 8.25 at the fifth. Even though the differences are relatively large, they are only statistically significant at prefixes two and five (with 2.4 and 4 percent relative improvement over MAE, respectively). Looking at figure 5, the difference between the MAE_{EtD} seems to increase as the prefix gets larger, while the spread of the MAE-loss performance also increases for larger prefixes.

Finally, for the Hospital billing data, the performance of the temporal loss functions is again better than the baseline across all five prefixes, however, with less than one day of difference (except at prefix one) and statistically insignificant. In the Hospital billing data, the MAE_{EtD} was again performing the best most of the time.

Temporal consistency

The performance results in terms of the *average increase in predicted remaining time*, measured as the Temporal Consistency, can be seen from figure 6, and tables 9 and 11. The results show that the MAE baseline loss function has the best temporal consistency across all four event-logs, where the MAE_{EtD} has the worst temporal consistency across all but the Hospital billing data. The differences across the temporal loss functions and the baseline are, however, lower than a single day on average. Pairwise F-test for the difference between each temporal loss and the baseline (MAE) was also performed and can be found in table 10. The results of these tests show that the differences reported in table 9 are not significantly different.

Looking at the results in figure 6, it can be seen that the MAE-loss most often has the lowest TC_t across the event-logs, not only in the earliest/shortest prefixes but also for the longest, especially for the Sepsis data. However, F-test was performed for the results in table 11, and only the MAE_{PtD} at prefix two for the Helpdesk data, had a significant difference from the results of the MAE-loss. Looking at the general trends in 6, it can be observed that there are certain *peak* periods where prediction uncertainty is generally higher, and the predicted remaining time seems to increase (prefix two in the Sepsis data, six and nine in Traffic fines data, as well as six and ten in Hospital billing data). For the Traffic fines data specifically, the increased predicted remaining time can become as much as three to four months across all four loss functions.

Discussion

As the aim of the temporal loss functions was to bias the trained models towards a better *earliness* by prioritizing this perspective higher than the standard MAE loss, it was unexpected that this would lead to better accuracy than the MAE. In a previous study [1], the relationship between earliness and accuracy has been referred to as a trade-off, which does not seem to be the case for the performed experiments in this study.

However, viewing the results from the perspective of the proposed temporal consistency metric, there might be another trade-off between accuracy and earliness on one side, and the temporal consistency on the other. Even though the proposed loss functions produce better average accuracy, and in some cases earliness, the resulting models do on average also seem to perform worse in terms of temporal consistency even though the differences are small and often insignificant. This furthermore seems to be a general problem across all four loss functions.

Looking at figure 6, there are significant peaks in the increase of predicted remaining time (TC_t) at some prefixes. This problem seems to be worst for the Traffic fines and Hospital billing data. Looking at figure 3, one can also observe that the prefix distributions have significant cut-off points at prefixes five and six, in both train and test periods for these event-logs. The *peaks* in TC errors might therefore be related to the lower support for these prefix lengths. From a planning perspective, it might be useful to know these weaknesses of the remaining time models, before they are used operationally. Future work might therefore focus on enhancing performance in this area.

Going back to the relative performance of the proposed loss functions, statistically significant improvements were found for some of the evaluated event-log data with up to 5.24 days improvement at prefix two in the Traffic fines data. On average, the temporal loss functions performed better in terms of earliness, but only significantly so in the Sepsis and Traffic fines data.

As it was found that the MAE_{EtD} had the best earliness in most cases, it would suggest that the curvature of the temporal decay plays an important role, and that the formulation of the MAE_{MtD} and MAE_{PtD} losses are too extreme for the evaluated settings. However, as the optimal hyper-parameters also differ for each of the four included event-logs, it is speculated that there might be situations where higher temporal decay would perform better. The interaction between the peak and slope of the temporal decay, as well as distribution characteristics of the event-logs should therefore be studied further in the future. The general learning from this study is therefore that the temporal decay could be used as a hyper-parameter, which might lead to better earliness performance in some situations. On the other hand, it is now known that this also can lead to a decrease in temporal consistency. This apparent trade-off motivates the need to evaluate remaining time models from the perspectives of accuracy, earliness, and in some cases temporal consistency when used for operational decision support.

Conclusion and future research

The aim of this study was to investigate the impact of temporally weighted loss functions for remaining time prediction from event-log data. This was achieved by adding a temporal decay to the standard *MAE* loss function used in previous studies, aiming at improving earliness performance. Furthermore, a previously untested aspect of model performance was proposed: Temporal consistency, which measures the degree to which the predictions follow the natural monotonic behavior of remaining time.

The results show that performance improvements in terms of model earliness can be achieved using the proposed MAE_{EtD} in two of the four evaluated event-logs. Average improvements were up to 4% within the first five prefixes of the Traffic fines data. As the improvement is only statistically significant for two of four event-logs, it is argued that the temporal loss functions should be treated as a hyper-parameter, as the optimal peak and slope of the loss might be different depending on the distribution of the training data and the remaining hyper-parameters. Future works might therefore further study the relationship between the curvature of the loss and distribution characteristics of the training data, for instance via simulation.

Furthermore, it is found that across all the evaluated loss functions, peaks in the measured temporal consistency error might appear at certain time steps. These vulnerabilities in the models might be problematic when used for dynamic planning such as job scheduling, and the temporal consistency might therefore be a useful addition to accuracy and earliness in these cases. As temporal weighting of the loss functions seems to improve performance in the remaining time prediction domain, future research might study the effect of temporal weighting for classification problems such as next activity or case outcome prediction.

Conflict of interest

The author states no conflict of interest.

References

1. Verenich I, Dumas M, Rosa ML, Maggi FM, and Teinemaa I. Survey and Cross-benchmark Comparison of Remaining Time Prediction Methods in Business Process Monitoring. ACM Transactions on Intelligent Systems and Technology 2019; 10:1–34. DOI: 10.1145/3331449. arXiv: arXiv:1805.02896v2
2. Kubrak K, Milani F, Nolte A, and Dumas M. Prescriptive process monitoring: Quo vadis? PeerJ Computer Science 2022; 8:e1097
3. Teinemaa I, Dumas M, Maggi F, and Di Francescomarino C. Predictive Business Process Monitoring with Structured and Unstructured Data. 2016 Sep :401–17. DOI: 10.1007/978-3-319-45348-4_23
4. Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa. Predictive Business Process Monitoring with LSTM Neural Networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2017; 10253 LNCS:V–VI. DOI: 10.1007/978-3-319-59536-8
5. Navarin N, Vincenzi B, Polato M, and Sperduti A. LSTM networks for data-aware remaining time prediction of business process instances. 2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings 2018; 2018-Janua:1–7. DOI: 10.1109/SSCI.2017.8285184. arXiv: arXiv:1711.03822v1
6. Hsu SY and Liu CH. Improving the delivery efficiency of the customer order scheduling problem in a job shop. Computers & Industrial Engineering 2009; 57:856–66
7. Averbakh I and Baysan M. Approximation algorithm for the on-line multi-customer two-level supply chain scheduling problem. Operations Research Letters 2013; 41:710–4
8. Testi A, Tanfani E, and Torre G. A three-phase approach for operating theatre schedules. Health care management science 2007; 10:163–72
9. Lujak M and Billhardt H. A Distributed Algorithm for Dynamic Break Scheduling in Emergency Service Fleets. *PRIMA 2017: Principles and Practice of Multi-Agent Systems*. Ed. by An B, Bazzan A, Leite J, Villata S, and Torre L van der. Cham: Springer International Publishing, 2017 :477–85
10. Valouxis C and Housos E. Hybrid optimization techniques for the workshift and rest assignment of nursing personnel. Artificial intelligence in medicine 2000 Nov; 20:155–75. DOI: 10.1016/S0933-3657(00)00062-2
11. Wang TC and Liu CC. Optimal Work Shift Scheduling with Fatigue Minimization and Day Off Preferences. Mathematical Problems in Engineering 2014 Apr; 2014. DOI: 10.1155/2014/751563
12. Cheng MY, Huang KY, and Hutomo M. Multiobjective Dynamic-Guiding PSO for Optimizing Work Shift Schedules. Journal of Construction Engineering and Management 2018 Sep; 144. DOI: 10.1061/(ASCE)CO.1943-7862.0001548
13. Dumas M, Rosa ML, Mendling J, and Reijers HA. Fundamentals of Business Process Management. 2nd. Springer Publishing Company, Incorporated, 2018

14. Rama-Maneiro E, Vidal J, and Lama M. Deep Learning for Predictive Business Process Monitoring: Review and Benchmark. ArXiv 2020; abs/2009.13251

15. Metzger A, Neubauer A, Bohn P, and Pohl K. Proactive process adaptation using deep learning ensembles. *International Conference on Advanced Information Systems Engineering*. Springer. 2019 :547–62

16. Di Francescomarino C, Dumas M, Federici M, Ghidini C, Maggi FM, Rizzi W, and Simonetto L. Genetic Algorithms for Hyperparameter Optimization in Predictive Business Process Monitoring. Inf. Syst. 2018 May; 74:67–83. DOI: 10.1016/j.is.2018.01.003

17. Teinemaa I, Dumas M, Leontjeva A, and Maggi FM. Temporal stability in predictive process monitoring. Data Mining and Knowledge Discovery 2018; 32:1306–38. DOI: 10.1007/s10618-018-0575-9. arXiv: arXiv:1712.04165v3

18. Thompson B, Stock M, Banuelas V, and Akalonu C. The Impact of a Rigorous Multiple Work Shift Schedule and Day Versus Night Shift Work on Reaction Time and Balance Performance in Female Nurses: A Repeated Measures Study. Journal of Occupational and Environmental Medicine 2016 May; 58:1. DOI: 10.1097/JOM.0000000000000766

19. Fido A and Ghali A. Detrimental Effects of Variable Work Shifts on Quality of Sleep, General Health and Work Performance. Medical principles and practice : international journal of the Kuwait University, Health Science Centre 2008 Oct; 17:453–7. DOI: 10.1159/000151566

20. Bard J and Purnomo H. Short-Term Nurse Scheduling in Response to Daily Fluctuations in Supply and Demand. Health care management science 2005 Dec; 8:315–24. DOI: 10.1007/s10729-005-4141-9

21. Van der Aalst WMP. Process Mining: Data Science in Action. 2nd ed. Heidelberg: Springer, 2016. DOI: 10.1007/978-3-662-49851-4

22. Dumas M, Rosa ML, Mendling J, and Reijers HA. Fundamentals of Business Process Management. Springer Publishing Company, Incorporated, 2013

23. Van der Aalst WM. Process discovery from event data: Relating models and logs through abstractions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2018; 8:e1244. DOI: 10.1002/widm.1244

24. Goodfellow I, Bengio Y, and Courville A. Deep Learning. The MIT Press, 2016

25. Hochreiter S and Schmidhuber J. Long Short-term Memory. Neural computation 1997 Dec; 9:1735–80. DOI: 10.1162/neco.1997.9.8.1735

26. Hochreiter S, Bengio Y, Frasconi P, and Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by Kremer SC and Kolen JF. IEEE Press, 2001

27. Mannhardt F, Leoni M de, Reijers HA, and Aalst WMP van der. Data-Driven Process Discovery - Revealing Conditional Infrequent Behavior from Event Logs. *Advanced Information Systems Engineering*. Ed. by Dubois E and Pohl K. Cham: Springer International Publishing, 2017 :545–60

28. Mannhardt F and Blinde D. Analyzing the Trajectories of Patients with Sepsis using Process Mining. 2017 Jun

29. Mannhardt F, Leoni M de, Reijers H, and Aalst W. Balanced multi-perspective checking of process conformance. Computing 2015 Feb. DOI: 10.1007/s00607-015-0441-1

30. Verenich I. Helpdesk. Mendeley data 2016 Dec. DOI: 10.17632/39bp3vv62t.1

31. LeCun Y, Bottou L, Orr G, and Müller K. Efficient BackProp. *Neural Networks: Tricks of the Trade*. 1998

32. Camargo M, Dumas M, and González-Rojas O. Learning Accurate LSTM Models of Business Processes. 2019 Jul :286–302. DOI: 10.1007/978-3-030-26619-6_19

33. Dozat T. Incorporating Nesterov Momentum into Adam. *Proceedings of the 4th International Conference on Learning Representations*. 2016 :1–4

34. Van Dongen BF, Crooy RA, and Van Der Aalst WMP. Cycle time prediction: When will this case finally be finished? Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2008; 5331 LNCS:319–36. DOI: 10.1007/978-3-540-88871-0_22

35. Khan A, Le H, Do K, Tran T, Ghose A, Dam H, and Sindhgatta R. Memory-Augmented Neural Networks for Predictive Process Analytics. arXiv preprint arXiv:1802.00938 2018. arXiv: 1802.00938. Available from: http://arxiv.org/abs/1802.00938

36. Evermann J, Rehse JR, and Fettke P. A deep learning approach for predicting process behaviour at runtime. International Conference on Business Process Management 2016; 1:490. DOI: 10.1007/978-3-319-58457-7. Available from: http://b-ok.xyz/book/2942192/1d94cd

37. Feurer M and Hutter F. Hyperparameter Optimization. *Automated Machine Learning: Methods, Systems, Challenges*. Ed. by Hutter F, Kotthoff L, and Vanschoren J. Cham: Springer International Publishing, 2019 :3–33. DOI: 10.1007/978-3-030-05318-5_1. Available from: https://doi.org/10.1007/978-3-030-05318-5_1

38. Verenich I, Nguyen H, Rosa ML, and Dumas M. White-box prediction of process performance indicators via flow analysis. *ACM International Conference Proceeding Series*. Vol. Part F128767. 2017. DOI: 10.1145/3084100.3084110

39. Senderovich A, Di Francescomarino C, Ghidini C, Jorbina K, and Maggi FM. Intra and Inter-case Features in Predictive Process Monitoring: A Tale of Two Dimensions. *Business Process Management*. Ed. by Carmona J, Engels G, and Kumar A. Cham: Springer International Publishing, 2017 :306–23

40. LeCun Y, Bottou L, Orr GB, and Müller K. Efficient BackProp. *Neural Networks: Tricks of the Trade*. Ed. by Orr GB and Müller KR. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998 :9–50. DOI: 10.1007/3-540-49430-8_2

41. Hastie T, Tibshirani R, and Friedman J. The Elements of Statistical Learning. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001

42. Gers FA, Schraudolph NN, and Schmidhuber J. Learning Precise Timing with Lstm Recurrent Networks. J. Mach. Learn. Res. 2003 Mar; 3:115–43. DOI: 10.1162/153244303768966139

43. Appleyard J, Kociský T, and Blunsom P. Optimizing Performance of Recurrent Neural Networks on GPUs. ArXiv 2016; abs/1604.01946

44. Wedel M, Hacht M, Hieber R, Metternich J, and Abele E. Real-time Bottleneck Detection and Prediction to Prioritize Fault Repair in Interlinked Production Lines. Procedia CIRP 2015 Dec; 37:140–5. DOI: 10.1016/j.procir.2015.08.071

45. Chollet F. Deep Learning with Python. Manning, 2017

46. Francescomarino CD, Dumas M, Maggi FM, and Teinemaa I. Clustering-Based Predictive Process Monitoring. 2015. arXiv: 1506.01428 [cs.SE]

47. Senderovich A, Di Francescomarino C, Ghidini C, Jorbina K, and Maggi F. Intra and Inter-case Features in Predictive Process Monitoring: A Tale of Two Dimensions. 2017 Aug :306–23. DOI: 10.1007/978-3-319-65000-5_18

48. Jan ST, Ishakian V, and Muthusamy V. AI Trust in Business Processes: The Need for Process-Aware Explanations. Proceedings of the AAAI Conference on Artificial Intelligence 2020 Apr; 34:13403–4. DOI: 10.1609/aaai.v34i08.7056. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/7056

49. Kingma D and Ba J. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations 2014 Dec

50. Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia Y, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. 2015. Available from: http://tensorflow.org/

51. Lundberg SM and Lee SI. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems 30*. Ed. by Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, and Garnett R. Curran Associates, Inc., 2017 :4765–74. Available from: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

52. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, and Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 2014; 15:1929–58. Available from: http://jmlr.org/papers/v15/srivastava14a.html

Operation	**Description**
Feature engineering	The event activity, resource and timestamps were used as features across all four event-logs. Similar to [17], additional time-related features such as the day of week and the hour of the day were encoded as categorical inputs. Apart from these, all other available features from the publicly available event-logs were used.
Standardization	To accommodate the most optimal settings for the back-propagation algorithm used for training the neural networks models in this study, numerical features are standardized as recommended in [31].
Exclusion of last event	As many event-logs do not include the timestamp for the end of a task, the standard formulation of remaining time models is thereby to predict time to the last event. As the remaining time at the last event is trivial to predict, it is common to drop the last event [5, 1].
Exclusion of cases with only one event	As the last event is dropped, there would be no events to predict the remaining time from. These cases are therefore excluded.
Prefix-log transformation	Using the same approach as in [4, 5, 17, 32] a prefix log is generated: Each event is transformed into a *prefix* consisting of all previous events in the *i'th* trace up until the *t'th* event, for all events.
Prefix-log truncation	As the tensor-based models used in this study need input tensors of a fixed size; truncation has been applied as in [17] (see section: data). Leading zero-padding have been applied across all event-logs.
Tensor-encoding of prefixes	The final prefix log have been transformed into an input tensor with 3 axes *(cases x prefixes x features)* as described in [1]. The target values are represented by a single vector.

Table 4: *Data preparation of each experiment.*

Event-log	Loss	Units	Layers	Alg	BS	LR
Sepsis	*MAE*	100	2	NADAM	1024	0.10
Helpdesk	*MAE*	100	1	SGD	2048	0.10
Traffic fines	*MAE*	100	1	NADAM	128	0.10
Hospital billing	*MAE*	100	2	NADAM	128	0.01

Table 5: Hyper-parameters used across main experiments. Loss: loss function, Units: number of hidden units, Layers: number of LSTM-layers, Alg: Optimization algorithm, LR: Learning rate.

Loss function	Sepsis	Helpdesk	Traffic fines	Hospital billing
MAE	12.39 ± 0.11	8.99 ± 0.33	214.13 ± 3.55	46.55 ± 0.61
MAE_{EtD}	**12.33** ± 0.05	8.63 ± 0.34	**208.91** ± 1.77	46.34 ± 0.57
MAE_{PtD}	12.45 ± 0.12	8.81 ± 0.53	210.21 ± 2.52	46.39 ± 0.55
MAE_{MtD}	12.34 ± 0.08	**8.59** ± 0.29	209.99 ± 3.35	**46.24** ± 0.46

Table 6: Average MAE of each loss function across event-logs, measured in days. RHS of ± denotes the 95% confidence intervals.

Event-log	Loss function	F-value	P-value
Sepsis	MAE_{EtD}	3.938	***** 0.026
Sepsis	MAE_{PtD}	0.750	0.662
Sepsis	MAE_{MtD}	1.642	0.235
Helpdesk	MAE_{EtD}	0.902	0.559
Helpdesk	MAE_{PtD}	0.382	0.915
Helpdesk	MAE_{MtD}	1.305	0.349
Traffic fines	MAE_{EtD}	4.027	***** 0.024
Traffic fines	MAE_{PtD}	1.978	0.161
Traffic fines	MAE_{MtD}	1.117	0.435
Hospital billing	MAE_{EtD}	1.137	0.425
Hospital billing	MAE_{PtD}	1.253	0.370
Hospital billing	MAE_{MtD}	1.745	0.209

Table 7: F-test results of pairwise comparison between the three temporal loss functions and the MAE-loss baseline. '*****' denote significant difference at $\alpha = 0.05$.

Prefix	Loss function	Sepsis	Helpdesk	Traffic fines	Hospital billing
1	MAE	13.27 ± 0.21	7.07 ± 0.05	227.46 ± 3.66	70.2 ± 1.22
	MAE_{EtD}	**13.14** ± 0.2	**7.06** ± 0.16	**222.11** ± 2.36	70.0 ± 1.38
	MAE_{PtD}	13.35 ± 0.24	7.08 ± 0.14	224.26 ± 4.43	69.31 ± 1.24
	MAE_{MtD}	13.17 ± 0.22	**7.06** ± 0.16	225.24 ± 5.2	**69.12** ± 1.07
2	MAE	13.0 ± 0.17	8.52 ± 0.23	217.32 ± 3.93	45.37 ± 0.56
	MAE_{EtD}	**12.91*** ± 0.09	8.39 ± 0.21	**212.08*** ± 2.08	45.23 ± 0.43
	MAE_{PtD}	13.08 ± 0.2	8.31 ± 0.31	213.2 ± 2.64	**45.13** ± 0.49
	MAE_{MtD}	12.94 ± 0.13	**8.29** ± 0.23	212.22 ± 3.73	45.3 ± 0.39
3	MAE	13.52 ± 0.16	9.68 ± 0.37	218.67 ± 4.33	42.23 ± 0.54
	MAE_{EtD}	**13.43*** ± 0.08	9.4 ± 0.36	**212.99** ± 3.35	**41.99** ± 0.45
	MAE_{PtD}	13.59 ± 0.2	9.53 ± 0.62	213.49 ± 3.55	42.21 ± 0.54
	MAE_{MtD}	13.47 ± 0.13	**9.33** ± 0.43	213.2 ± 4.05	42.18 ± 0.55
4	MAE	13.5 ± 0.15	12.72 ± 1.22	198.79 ± 4.71	39.32 ± 0.45
	MAE_{EtD}	**13.43*** ± 0.07	**10.88** ± 1.34	**195.22** ± 3.77	**39.0** ± 0.41
	MAE_{PtD}	13.58 ± 0.2	12.14 ± 1.87	196.2 ± 4.5	39.56 ± 0.44
	MAE_{MtD}	13.47 ± 0.12	11.06 ± 0.96	195.39 ± 4.5	39.3 ± 0.45
5	MAE	14.37 ± 0.15	12.93 ± 1.39	201.82 ± 16.99	42.14 ± 0.41
	MAE_{EtD}	**14.3*** ± 0.07	**11.19** ± 1.5	**193.57*** ± 1.58	**42.1** ± 0.32
	MAE_{PtD}	14.44 ± 0.2	12.27 ± 1.86	196.25***** ± 5.04	42.42 ± 0.37
	MAE_{MtD}	14.33 ± 0.13	11.29***** ± 0.76	200.39 ± 16.97	42.19 ± 0.28

Table 8: Average MAE_t of loss functions in prefixes $t = (1, ..., 5)$, measured in days. '*****' denote significant difference at $\alpha = 0.05$, for a pairwise F-test between each temporal loss function and the MAE-loss baseline. Each pairwise comparison is computed within each prefix-length only.

Loss function	Sepsis	Helpdesk	Traffic fines	Hospital billing
MAE	**0.17** ± 0.02	**0.272** ± 0.1	**2.137** ± 0.92	**1.982** ± 0.1
MAE_{EtD}	0.189 ± 0.02	0.427 ± 0.13	3.004 ± 1.18	2.07 ± 0.23
MAE_{PtD}	0.207 ± 0.03	0.302 ± 0.2	2.447 ± 0.7	2.162 ± 0.25
MAE_{MtD}	0.203 ± 0.03	0.408 ± 0.19	2.768 ± 1.74	2.032 ± 0.14

Table 9: Average TC of different loss functions across event-logs, measured in days. RHS of ± denotes the 95% confidence intervals.

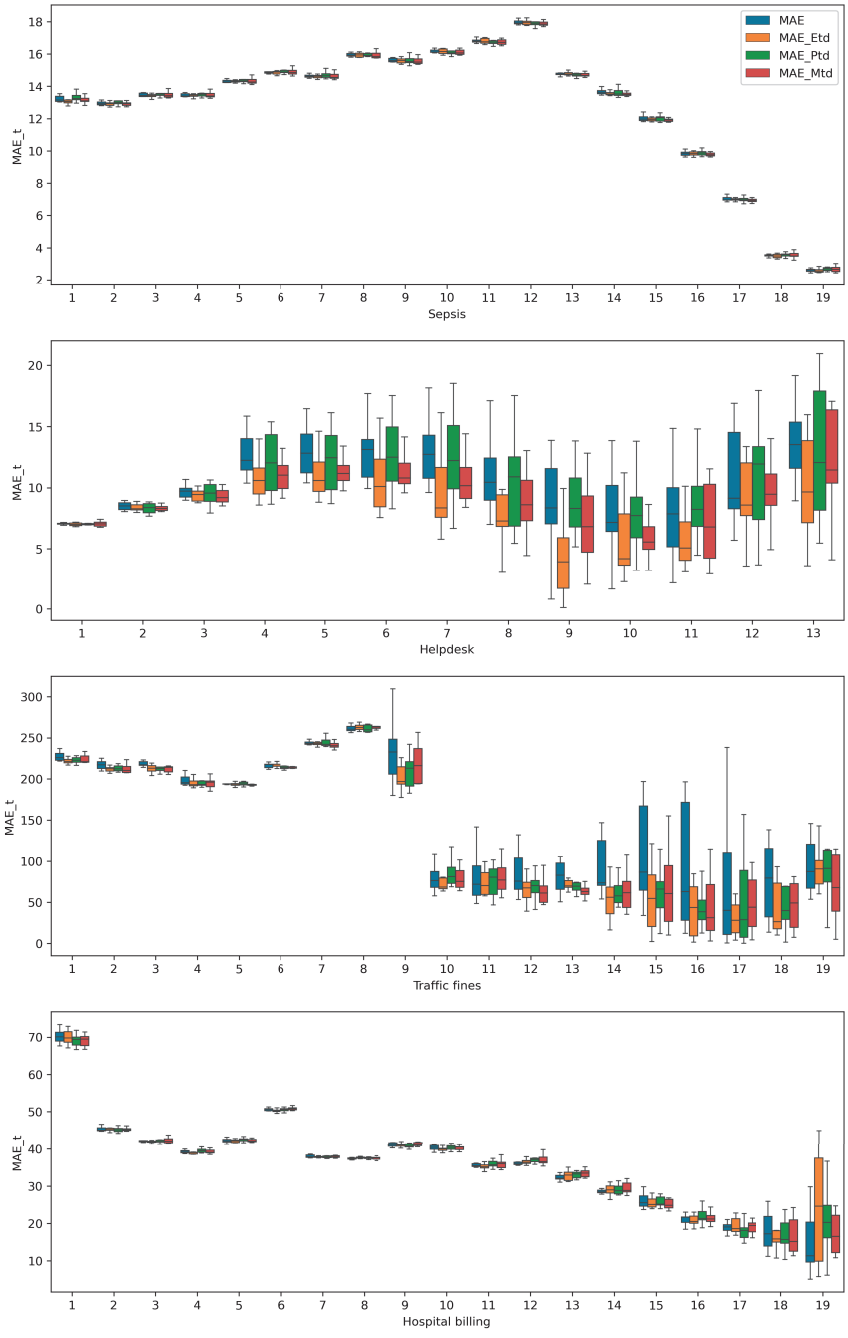


Figure 5: MAE_t at the first 5 events, across all traces in each event-log, measured in days.

Event-log	Loss function	F-value	P-value
Sepsis	MAE_{EtD}	1.206	0.393
Sepsis	MAE_{PtD}	0.576	0.788
Sepsis	MAE_{MtD}	0.720	0.684
Helpdesk	MAE_{EtD}	0.653	0.733
Helpdesk	MAE_{PtD}	0.266	0.969
Helpdesk	MAE_{MtD}	0.300	0.956
Traffic fines	MAE_{EtD}	0.613	0.761
Traffic fines	MAE_{PtD}	1.756	0.207
Traffic fines	MAE_{MtD}	0.281	0.964
Hospital billing	MAE_{EtD}	0.187	0.990
Hospital billing	MAE_{PtD}	0.154	0.995
Hospital billing	MAE_{MtD}	0.512	0.833

Table 10: F-test results of pairwise comparison between the three temporal loss functions and the MAE-loss baseline for average TC as defined by equation 13. '*****' denote significant difference at $\alpha = 0.05$.

Prefix	Loss function	Sepsis	Helpdesk	Traffic fines	Hospital billing
2	MAE	0.55 ± 0.07	0.3 ± 0.21	**0.47** ± 0.48	2.18 ± 0.38
	MAE_{EtD}	**0.54** ± 0.17	0.31 ± 0.19	1.54 ± 1.54	2.06 ± 0.33
	MAE_{PtD}	0.6 ± 0.2	**0.14*** ± 0.12	0.57 ± 0.72	**2.04** ± 0.38
	MAE_{MtD}	0.55 ± 0.11	0.33 ± 0.19	1.05 ± 0.65	2.09 ± 0.42
3	MAE	**0.21** ± 0.06	**0.27** ± 0.11	**2.53** ± 1.68	**1.16** ± 0.21
	MAE_{EtD}	0.26 ± 0.07	0.61 ± 0.2	3.33 ± 1.61	1.38 ± 0.26
	MAE_{PtD}	0.28 ± 0.07	0.52 ± 0.39	3.56 ± 1.96	1.34 ± 0.3
	MAE_{MtD}	0.24 ± 0.07	0.51 ± 0.28	3.77 ± 4.7	1.29 ± 0.2
4	MAE	0.12 ± 0.03	0.23 ± 0.07	2.13 ± 1.42	**1.5** ± 0.22
	MAE_{EtD}	**0.12** ± 0.02	0.25 ± 0.1	2.63 ± 2.55	1.51 ± 0.26
	MAE_{PtD}	0.15 ± 0.03	**0.17** ± 0.11	**1.81** ± 1.05	1.79 ± 0.34
	MAE_{MtD}	0.13 ± 0.03	0.3 ± 0.17	2.2 ± 1.79	1.57 ± 0.25
5	MAE	**0.1** ± 0.02	**0.25** ± 0.12	2.32 ± 0.61	**2.08** ± 0.34
	MAE_{EtD}	0.11 ± 0.03	0.53 ± 0.17	2.61 ± 0.72	2.55 ± 0.74
	MAE_{PtD}	0.13 ± 0.03	0.3 ± 0.21	**2.1** ± 1.0	2.7 ± 0.88
	MAE_{MtD}	0.12 ± 0.03	0.55 ± 0.25	3.82 ± 2.71	2.29 ± 0.44
6	MAE	**0.1** ± 0.03	**0.3** ± 0.09	59.68 ± 18.26	6.4 ± 0.85
	MAE_{EtD}	0.13 ± 0.03	0.57 ± 0.2	68.99 ± 11.26	6.14 ± 0.71
	MAE_{MtD}	0.13 ± 0.03	0.71 ± 0.33	**52.58** ± 14.37	**6.09** ± 1.01
	MAE_{PtD}	0.13 ± 0.04	0.55 ± 0.32	62.28 ± 17.65	6.36 ± 0.99

Table 11: Average TC_t of the evaluated loss functions at prefixes $t = (2, ..., 6)$, measured in days ($t = 1$ is left out due to the formulation in equation 14). '*****' denote significant difference at $\alpha = 0.05$, for a pairwise F-test between each temporal loss function and the MAE-loss baseline. Each pairwise comparison is computed within each prefix-length only.

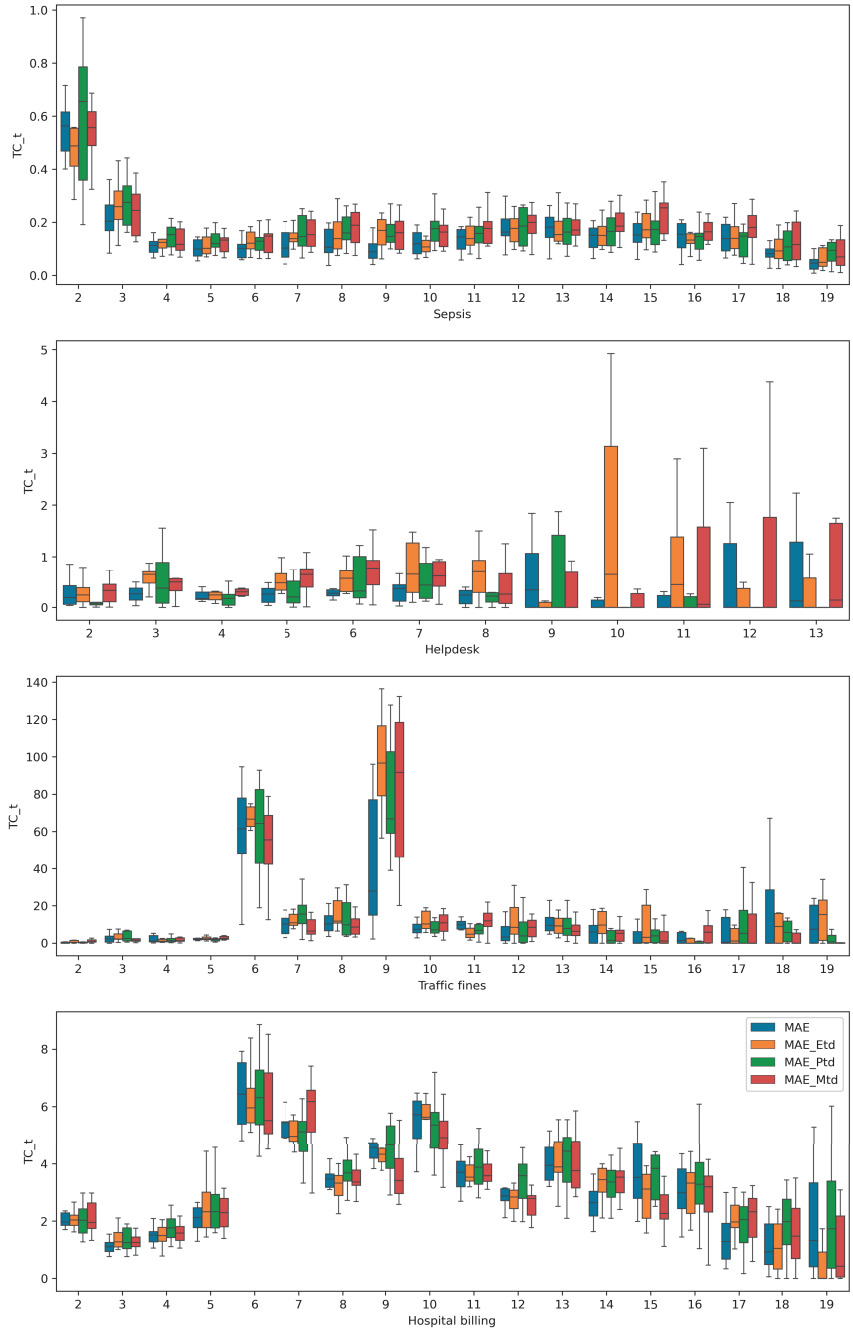


Figure 6: Average TC_t of loss functions at each prefix, measured in days ($t = 1$ is left out due to the formulation in equation 14).

Paper II

Riess, M. (2023a). A parametric simulation framework for the generation of event-log data. Manuscript submitted to *Simulation*.

A parametric simulation framework for the generation of event-log data

Simulation
XX(X):1–23

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Mike Riess

Abstract

In the pursuit of ecological validity, current Business Process Simulation methods are calibrated from data of existing processes. This is important for realistic *what-if* analysis of an existing business process. However, this is not always the *right tool for the job*. To test certain hypotheses in the field of Predictive Process Monitoring, it will be more helpful to simulate event-log data from a theoretical process, where all aspects can be manipulated. One example is when assessing the influence of process complexity or variability, on the performance of a given new prediction method. In this case, the ability to include control variables and systematically change process characteristics is key to fully understanding their influence. Calibrating a simulation model from observed data alone can in these cases be limiting. This paper propose a simulation framework for the generation of synthetic event-log data, where aspects such as process complexity, stability, trace distribution, duration distribution and case arrivals can be fully controlled by the user. The overall architecture is described in detail, and a demonstration of the framework is presented.

Keywords

Event-log data, discrete event simulation, predictive process monitoring.

Introduction

In the field of Predictive Process Monitoring, multiple approaches have been proposed for the prediction of attributes from event-log data van der Aalst (2016) that describe the future state of a business process; next activity Evermann et al. (2016); Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa (2017), time to the next activity Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa (2017), remaining throughput time (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017; Navarin et al. 2018) and case outcome Teinemaa et al. (2018). Common to all prior research in this area is that proposed methods are evaluated across several event-logs sourced from companies or public institutions made publicly available through conferences (La Rosa and Soffer 2013; Teniente and Weidlich 2018) or publications (Mannhardt and Blinde 2017; Mannhardt et al. 2015). The key argument for such an approach is ecological validity, as the prediction model can be evaluated empirically using historical data from the environments it is intended to function. Furthermore, performance across multiple domains can be assessed and a common *baseline* can be established for comparison across studies. This can also be referred to as *benchmarking*, and these publicly available datasets are commonly referred to as *benchmark data* in the Machine learning community.

However, for Predictive Process Monitoring, strictly speaking, the unit of observation is a *process* or an organization. As processes will vary within each domain, a single *benchmark dataset* thereby only represents one variant. When some studies use three datasets or less (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017; Navarin et al. 2018), the external validity of the findings (beyond the performance comparison to the established baseline) thereby suffer. If an objective is to understand the influence of factors related to the data itself, an experimental design with control variables will be needed.

As an example, the authors in (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017) found that their proposed prediction model performed relatively worse in one of two benchmark datasets it was evaluated on. Further investigating the data, where performance was worse, the authors found that the data had many sequences

Corresponding author:
Mike Riess, School of Economics and Business, Norwegian University of Life Sciences, Universitetstunet 3 1433 Ås, NO.
Email: mike.riess@nmbu.no

with multiple instances of the same activity, and thereby concluded that this was the cause of the worse performance. To further investigate this hypothesis, a simulation model could have been used to systematically generate event-logs with varying degrees of repeating sequences, as well as relevant control variables.

When findings are based on qualitative analysis of benchmark data, there are unfortunately few (if any) possibilities to control aspects of the environment. This can lead to problems of internal validity, as possible confounding variables cannot be studied and/or eliminated. For instance, in (Camargo et al. 2019a), the authors included 9 different event-logs for the evaluation of their enhanced prediction approach. Each event-log was qualitatively coded in terms of its complexity in the control flow (the *graph* of possible transitions between activities over time), and the variability in the time between events. The complexity factor included the following levels: Simple (2/9), Medium (2/9), Complex (5/9), and the Variability factor: Steady (3/9) and Irregular (6/9).

In this case, the levels within each factor was unevenly distributed, and there were no overlap between the two factors such that interaction effects could be determined. This study was therefore limited to qualitative analysis, as finding benchmark data that fit certain criteria can be challenging. A contribution to the findings of (Camargo et al. 2019a), could have been an experimental design and synthetic event-logs generated from a clearly defined distributions for each of the two factors. Firstly, this could enable the analysis of interactions between the two factors, as well including other control variables such as number of traces, and finally repeating the experiment with new data.

It appear, that the inclusion of simulation-based results, as an aid in the evaluation of Predictive Process Monitoring methods might benefit the advancements in the field. However, simulation is by no means a new tool in business process related research. Business process simulation (BPS) is a well-established stream of research within the Process mining Aalst (2015) community, with multiple frameworks aiming at generating realistic event-logs which are empirically calibrated. However, as discussed, simulation models following specific criteria to test a hypothesis can be a valuable addition to the current approach of evaluation using benchmark data. This is, in fact, an old tradition in the Machine learning literature (Tibshirani 1996; Efron et al. 2004; Zou and Hastie 2005; Bradley and Henseler 2007) that could help in the advancement of predictive process monitoring approaches.

In this study, a literature review is initially conducted to understand current limitations in the available open source simulation frameworks for the generation of event-log data. Next, a simulation framework that addresses these limitations is proposed, and finally a demonstration of the capabilities is performed.

Research questions

The main goals of this study is to contribute to the advancement in Predictive Process Monitoring by understanding the current limitations in the existing simulation tools, as well as to propose a new framework to aid in hypothesis-testing of data-related factors as discussed in the introduction. To achieve this, the work will be guided by the following two research questions:

- **RQ1:** To which extend does current business process simulation frameworks support the requirements for model robustness assessment in Predictive process monitoring?
- **RQ2:** How can the limitations in current simulation frameworks be addressed within a new framework?

To answer **RQ1** a literature review on Business Process Simulation frameworks is performed, while comparing their capabilities and design philosophy to the requirements discussed in the introduction. To answer **RQ2**, a set of initial requirements will firstly be outlined, where after a simulation framework will be proposed and demonstrated.

Theoretical background

In the following, the most important theoretical concepts and terms used in the literature review and framework presentation will be introduced.

Event log data

Event log data refers to time-stamped pieces of information related to a single *case* or *process instance* in a business process. Event-log data is generated in process-aware information systems (PAIS) (van der Aalst 2016) such as Enterprise Resource Planning system (ERP) and Customer Relationship Management (CRM) systems. A example event-log from a hypothetical *issue to resolution* process is illustrated in Table 1. An event-log can be viewed as a hierarchical data structure, where the highest level is a log Θ, which consist of multiple traces Q, $\Theta = \{Q_1, ..., Q_n\}$. Each trace has a unique identifier c and consist of one or more events e_t, arranged in the chronological order in which they

occurred, $Q_c = \{e_1, ..., e_T\}$. An event e_t is a tuple consisting of attributes such as an *activity* a_t, a *timestamp* n_t, and in many cases a *resource* r_t or other relevant attributes $o_{(t,i)}$. In the example in Table 1 (going from left to right), the columns denote the case identifier c, the case topic $o_{(t,i)}$, activity a_t, timestamp at the beginning of the activity n_t, and finally the resource r_t.

Business process simulation

The main objective of business process simulation is most often to get a deeper understanding of a specific business process (what-if analysis) (Aalst 2015). A business process simulation model can either be generated manually by process experts using observation and qualitative analysis Dumas et al. (2018), or automatically using e.g. *process mining* techniques (van der Aalst 2016) to aid the derivation of a control flow and its simulation parameters. A business process simulation model generally consist of the following components (Aalst 2015):

- Arrival process
- Control flow
- Resources
- Activity durations

The *arrival* of cases is most often modeled using a *Poisson process*, but can be extended to a more complex process depending on the use-case. The *control-flow* denotes the graph that determine the order and probability of various transitions between activities over time. The *resources* (employees or machines) performing the *activities* also need to be represented by the model and depending on the assumptions, resources have specific intervals wherein they are able to process activities. If unavailable, a queuing mechanism is needed, such as first-come first-served (FCFS) (Aalst 2015). Finally, the processing time, or *activity duration* needs to be represented by the model following a given distribution. Multiple simulation *languages* and *packages* (Aalst 2015) exist to aid in business process simulation. Classical tools such as ARENA (Altiok and Melamed 2007) and CPN Tools (Ratzer et al. 2003) let the user manually specify the simulation components mentioned earlier.

Predictive process monitoring

The field of predictive process monitoring focus on predicting future states of a process instance (a trace), based on machine learning models trained from historical event-log data (Teinemaa et al. 2019). These predictions can in turn be used for prescriptive process monitoring, which aim to provide operational decision support (Kubrak et al. 2022) in order to improve process performance. In the literature, one of the most common attributes to predict is the remaining throughput time y_t of a trace (van Dongen et al. 2008), at any discrete event up to the final event: $y_t = n_T - n_t$, where n_T and n_t denote the final and current timestamp in the trace, respectively. The time to the next event y_{t+1} (or the duration of the current event) is also a common prediction task (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017), formulated as $y_{t+1} = n_{t+1} - n_t$, where n_{t+1} denote the timestamp of the next event. Furthermore, it is also common to predict future categorical attributes of the process such as the next activity a_{t+1} (Evermann et al. 2016; Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017), the final activity a_T, or any other case attribute in time $o_{(t,i)}$ such as a *case outcome* Teinemaa et al. (2019). Common for these prediction approaches is that they rely on a sequence of discrete events, structured in the event-log format described in the previous section. Similar to most other fields of machine learning, the research in predictive process monitoring is primarily open source and performed using the Python programming language. For an event-log simulator to benefit this community, these characteristics are therefore important requirements.

Markov processes

Markov processes are stochastic processes defined by *Markov chains* (Rubinstein and Kroese 2016). A first-order Markov chain is defined as a *memoryless* discrete-time process of transitions between a finite set of states S in a state space D, $S \in D$. A state at time $t+1$ is assumed to be dependent only on the previous state at time t:

$$P(X_{t+1} \mid X_1, X_2, ..., X_T) = P(X_{t+1} \mid X_t) \tag{1}$$

Where $\{X_1, ..., X_T\}$ denote an arbitrary sequence of previously observed states. The initial probabilities of each state S_i in the state space D is given by the vector P^0. Transitions between states are thereby represented by the transition matrix P, which is a $|D| \times |D|$ matrix for a first-order $(k = 1)$ Markov chain. This will be referred to as a *memoryless* process in the following.

$$P^0 = \begin{bmatrix} P^0_{(1)} \\ \vdots \\ P^0_{(i)} \\ \vdots \\ P^0_{(D)} \end{bmatrix} \tag{2}$$

Case ID	Case topic	Activity	Timestamp	Resource
1001	Invoice	Email interaction	01-01-2019 15:01	System
1001	Invoice	Phone interaction	01-01-2019 16:04	User 2
1001	Invoice	Close case	01-01-2019 16:58	System
1002	Support	Email interaction	01-01-2019 12:01	System
1002	Support	Phone interaction	01-01-2019 13:10	User 2
1002	Support	Email interaction	01-01-2019 14:15	User 5
1002	Support	Close case	02-01-2019 13:37	System

Table 1. *Example event-log in a customer service unit.*

$$P = \begin{bmatrix} P_{(11)} & \dots & P_{(1j)} & \dots & P_{(1D)} \\ \vdots & \ddots & \vdots & & \vdots \\ P_{(i1)} & \dots & P_{(ij)} & \dots & P_{(iD)} \\ \vdots & & \vdots & \ddots & \vdots \\ P_{(D1)} & \dots & P_{(Dj)} & \dots & P_{(DD)} \end{bmatrix} \quad (3)$$

Each column j in P represent the probability of *j'th* state in the state space D at time $t+1$, given the current state represented by the *i'th* row in P. A Markov chain can be *absorbing* (Rubinstein and Kroese 2016), if it has an absorption state S_{END}, in addition to the ordinary (non-absorbing) states. If a sequence (trace) transitions into S_{END}, the sequence will end, as all transition probabilities from this state will be zero. For absorbing Markov chains, the absorbing state will be defined as the last row and column of the transition matrix.

Higher-order Markov chains

An extension of the *memoryless* Markov chain is the higher-order Markov chain (Raftery 1985; Ching et al. 2005), which assumes that the current state depend not only on the previous. Here, the next state e_{t+1} is conditioned on the previous k states:

$$X_{t+1} = P(X_{t+1} \mid X_1, X_2, ..., X_t) = P(X_{t+1} \mid X_t, X_{t-1}, ..., X_{t-k+1}) \quad (4)$$

Where k represents the order of the Markov chain, and thus how much memory it has (the number of previous states to condition the current state at time t on). As the amount of memory k grows, the number of parameters in the transition matrix increases exponentially. Hence, a *k'th* order Markov chain has $D \times D^k$ model parameters.

The transition probability matrix of a k-th order Markov chain can be represented as a first order Markov chain of $(k+1)$-tuples. This results in a state-set of size $|D|^k|D|$. A second-order Markov transition matrix can thus be illustrated as:

$$P^{(k=2)} = \begin{bmatrix} P^{(2)}_{(1,1,1)} & \dots & P^{(2)}_{(1,1,j)} & \dots & P^{(2)}_{(1,1,D)} \\ \vdots & \ddots & \vdots & & \vdots \\ P^{(2)}_{(i,i,1)} & \dots & P^{(2)}_{(i,i,j)} & \dots & P^{(2)}_{(i,i,D)} \\ \vdots & & \vdots & \ddots & \vdots \\ P^{(2)}_{(D,D,1)} & \dots & P^{(2)}_{(D,D,j)} & \dots & P^{(2)}_{(D,D,D)} \end{bmatrix} \quad (5)$$

Each cell in the transition matrix (5) correspond to the probability of a sequence of length $k+1$, rather than a single transition between two states. Using the upper-left cell as an example, this transition represents the following sequence in the state-space D: $D_1 \longrightarrow D_1 \longrightarrow D_1$, where the third state in this sequence is dependent on the previous two.

For a first-order Markov chain, the trace length is ultimately a function of the size of the state space D and the transition probabilities (number and likelihood of transitions between ordinary states and the absorbing state). Simulating an absorbing *memoryless* process, the sequence will continue to grow until the absorbing state is reached. This principle can also be used for higher-order Markov chains, where the prediction horizon is k, and the $k+1$*'th* state depend on the last k states as illustrated in Equation 4.

Poisson process

A Poisson process is a stochastic process describing a particular pattern of arrivals. Its key characteristics include (Rubinstein and Kroese 2016):

1. Arrivals $N_t = \{T_1, ..., T_k\}$ are countable.
2. Arrivals N_t occur in intervals: $N_1 = I_1 = (a, b], N_2 = I_2 = (b, c], ...$
3. Arrivals across intervals $\{I_1, ..., I_T\}$ are independent.
4. Arrival-times T_k within intervals $I_t = \{T_1, ..., T_k\}$ are independent and $Exp(\lambda)$-distributed, with λ being constant across all arrivals. Where $Exp(\lambda)$ denote the Exponential distribution.

If the above conditions are fulfilled, the probability of a single arrival in interval $I_k = (t, t+h]$ can be described

as $Poi(\zeta h) = e^{-\zeta h}\zeta h$, where ζ is the probability of an arrival, h is the window of observation, and $Poi(\zeta h)$ denote the Poisson distribution (presented in the next section). Intuitively, ζ can be seen as the *arrival-rate* parameter of a Poisson process.

Poisson distribution The probability mass function of the Poisson distribution is defined as:

$$f(k,\zeta) = P(N_t = k) = \frac{\zeta^k e^{-\zeta}}{k!} \tag{6}$$

Where k is the number of arrivals. For the Poisson distribution, the expectation ζ is also equal to the variance: $\zeta = E(N_t) = Var(N_t)$. Another characteristic of the Poisson process is that it has a direct link to the Exponential and Erlang distributions (described in the following), as: $Poi(\zeta n) = \sum_{i=1}^{n} Exp(\lambda) = Er(n,\lambda)$, when $n > 0 \in \mathbb{Z}$, and λ is a scalar.

Exponential distribution The exponential distribution (Rubinstein and Kroese 2016) is a memoryless continuous distribution in the range $x \in [0,\infty)$, using a single *rate* parameter λ. It represent the distribution of time between events in a Poisson process, and is a special case of the gamma distribution. The exponential distribution $Exp(\lambda)$ is memoryless in that the distribution of remaining time is the same at any point, independently of how much time has already passed. The probability density function (PDF) and cumulative distribution functions (CDF) are defined as:

$$PDF(x) = \lambda e^{-\lambda x} \qquad CDF(x) = 1 - e^{-\lambda x} \tag{7}$$

Expectation, variance:

$$E[X] = \frac{1}{\lambda}, \qquad VAR[X] = \frac{1}{\lambda^2} \tag{8}$$

Erlang distribution The Erlang distribution (Rubinstein and Kroese 2016) is a special case of the *hypoexponential* distribution (Trivedi and Bobbio 2017) (introduced in the next section) when all k states have an identical rate λ. In this case, the distribution is referred to as an Erlang distribution of order k. The Erlang distribution is thus a two-parameter family $Er(\tau,k)$.

The probability density function (PDF) and cumulative distribution function (CDF) is defined as:

$$PDF(x) = \frac{\lambda^k x^{k-1}}{(k-1)!} e^{-\lambda x} \tag{9}$$

$$CDF(X) = 1 - \sum_{i=0}^{k-1} \frac{(\lambda x)^i}{i!} e^{-\lambda x} \tag{10}$$

Expectation, variance:

$$E[X] = \frac{k}{\lambda}, \qquad VAR[X] = \frac{k}{\lambda^2} \tag{11}$$

Hypoexponential distribution The hypoexponential distribution (Trivedi and Bobbio 2017) is a combination of $k \geq 2$ sequential states which are exponentially distributed, but with individual rates $(\lambda_1, \lambda_2, ..., \lambda_k)$. A hypoexponential distribution of order $k = 2$, where $\lambda_1 \neq \lambda_2$ and $\lambda_1, \lambda_2 > 0$ is denoted as $Hypo(\lambda_1, \lambda_2)$. In this case, the distribution has the following probability density function:

$$PDF(x) = \frac{\lambda_1 \lambda_2}{\lambda_2 - \lambda_1}(e^{-\lambda_1 x} - e^{-\lambda_2 x}) \tag{12}$$

And cumulative density function:

$$CDF(x) = 1 - \frac{\lambda_2}{\lambda_2 - \lambda_1} e^{-\lambda_1 x} + \frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 x} \tag{13}$$

In the case of $Hypo(\lambda_1, ..., \lambda_k)$, the expected value, variance and coefficient of variation is defined as:

$$E[X] = \sum_{i=1}^{k} \frac{1}{\lambda_i}, \qquad VAR[X] = \sum_{i=1}^{k} \frac{1}{\lambda_i^2} \tag{14}$$

The expectation is the sum of k exponentially-distributed variables with individual rates, where the variance is the sum of the squared individual rates. The special case $Hypo(\lambda_1, \lambda_2)$, where $\lambda_2 = 0$ simply reduces to an exponential distribution with rate λ_1.

Discrete distributions

In the following, relevant discrete distributions used in the proposed framework will be briefly introduced.

Binomial distribution A binomial experiment is defined as more than one independent and identically distributed (i.i.d.) Bernoulli experiment. The binomial distribution is thus the sum of $(n > 1)$ Bernoulli experiments. The binomial distribution is often used to model the number of successes in n i.i.d. weighted coin tosses. The probability mass function of the binomial distribution can be described as:

$$PMF(k,n,p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{15}$$

Where k is the number of successes of n independent Bernoulli experiments with identical probability of success p. Similarly, the cumulative density function can be denoted as:

$$CDF(k,n,p) = \sum_{i=0}^{k} \binom{n}{i} (p)^i (1-p)^{n-i} \quad (16)$$

In the following, the expression $Binom(n,p)$ will denote inverse transform sampling of the cumulative density function of a binomial distribution (Fishman 2001).

Discrete Multinomial distribution The multinomial distribution is derived from the binomial distribution (Fishman 2001), as one or more experiments with multiple outcomes k. When $k = 2$ and number of trials $n = 1$, the multinomial distribution becomes the bernoulli distribution, and binomial when $n > 1$. When $k > 2$ it becomes the multinomial distribution which is specified by the following probability mass function:

$$PMF(k,n,p) = \frac{n!}{x_1! \dots x_k} p_1^{x_1} \dots p_k^{x_k} \quad (17)$$

Literature review

As the majority of recent research within Predictive Process Monitoring use the Python programming language as the standard for experiments and sharing of methods (Evermann et al. 2016; Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017; Navarin et al. 2018; Camargo et al. 2019a), the following literature review will be limited to free (open source) frameworks that are implemented in the Python language, such that they can be seamlessly integrated in future research projects.

The google scholar academic search engine was used to retrieve literature on existing simulation frameworks. The query used was: *"event-log simulation framework python"*, resulting in 1770 hits. Due to this large number of results, only studies with relevant keywords in the title or abstract were further reviewed. Cross references between frameworks in the literature review sections of the included studies were also used to discover relevant frameworks (leading to the inclusion of 1 additional study). The following inclusion criteria was formed for the found studies to qualify for further review: 1) The study presented a novel framework for the first time, 2) The framework was implemented in the python language, 3) The implementation was freely available online. A total of 9 frameworks met all criteria, listed in Table 2.

In the following, the found frameworks will be compared to the general criteria mentioned in the introduction of this paper. The found works have been classified in terms of 1) their general approach to simulation (empirical/calibrated and/or specified based on theory), 2) the components they include or allow the user to model, 3) the general type of simulation model used, and 4) distributions available (if applicable).

The majority of the found literature propose frameworks for generating a simulation model solely based on observed behaviour in the event-log. This will be referred to as an *empirical (E)* approach, where a hypothetical process generated by the user with no input data will be referred to as a *theoretical (T)* approach. Frameworks classified as both (E,T), include options to perform retrospective manipulations to the simulation model, after it has been inferred from the event-log.

The *SIMOD* framework (Camargo et al. 2019b) is split into processing and post-processing stages. In the processing stage the framework automatically generates a simulation model using SplitMiner (Augusto et al. 2017) to represent the control flow through a BMPN model and a set of probability distributions to model the remaining aspects of the process. Unfortunately, these are not further specified. In the post-processing step the framework optimizes the simulation model by changing the control flow in a manner that minimizes the distance between the simulated process and observed behaviour in the event-log.

PROSIMOS (López-Pintado and Dumas 2022) enhances the representation of resource behaviour by modelling the resource aspect in a similar way to what is classically known as agent-based simulation (Railsback et al. 2006), calibrated from the observed event-log. This framework does not produce the control flow and activity sequence distributions, but assume this to be an input generated from a framework such as SIMOD.

PNSIM (Pourbafrani et al. 2021b) extend the ability to generate a simulation model from observed behaviour in an event-log to performing changes to distributions of the activity durations and arrival rates. The control flow is represented by a Petri Net, and arrival times and activity durations are represented by distributions (which are not specified further). Contrary to the approach in PROSIMOS, the variation in and availability of resources are assumed to be represented by the duration distribution.

The *SIMPT* approach in (Pourbafrani et al. 2021a) generates a simulation model using process trees to represent the control flow and what is specified as *relevant distributions* to represent activity durations, waiting time, arrival rate and max capacity. This resulting simulation model is referred to as an *enriched Process Tree*. The next step of this framework is then for the user to specify the desired deviations to the simulation model, before running the simulation.

Source	Approach	Components	Control flow	Distributions
(Camargo et al. 2019b)	E	a, b, c, d	BPMN	Not documented
(López-Pintado and Dumas 2022)	E	a, b, c, d	BPMN	Not documented
(Pourbafrani et al. 2021b)	E, T	a, b, c	Petri net	Not documented
(Pourbafrani et al. 2021a)	E, T	a, b, c, d	Process tree	Not documented
(Fracca et al. 2021)	E, T	a, b, c, d	BPMN	Two documented: Poisson, Truncated Normal
(Pourbafrani and van der Aalst 2020)	E	a, b, c, d	CLD, SFD	Non-parametric (System Dynamics)
(Pegoraro et al. 2021)	E	b, c	Petri net	Not documented
(Grüger et al. 2022)	E, T	a, b, c	Data Petri net (DPN)	Not documented
(Peeperkorn et al. 2022)	E	a, b	LSTM-RNN	Non-parametric (LSTM)

Table 2. Comparison of found frameworks. Abbreviations: T: Theoretical, E: Empirical, a: Case arrivals, b: Activity sequences, c: Duration distributions, d: Resource availability, Mult: Multiple, BPMN: Business process model notation, CLD: Causal loop diagram, SFD: stock-flow diagram, SD: System Dynamics, LSTM-RNN: Long short-term memory recurrent neural network.

The *BPSIMPY* framework (Fracca et al. 2021) puts emphasis on the workflow management perspective, as the number of resources and their compatibility with task types are exclusively specified. Contrary to other approaches, this framework assumes an existing BPMN model of the process, and lets the user further specify scenarios and distribution parameters of the arrival rate, activity duration, waiting times and activity sequence. The proposed framework lists only two distributions in the documentation (in the example code), but more seem to be implemented.

PMSD proposed by Pourbafrani and van der Aalst (2020) uses systems dynamics (a statistical modelling technique) to represent an existing business process, using an event-log as the input. This approach is somewhat different from many of the other proposed frameworks, as the goal here is to create a system dynamics log, from which time series analysis can be performed. As the approach only includes functionality for calibration to observed dynamics in an existing process, this approach is purely empirical.

The *PROVED* framework (Pegoraro et al. 2021) aim primarily to perform so-called conformance checking (van der Aalst 2016), and thereby have little to no documentation on the simulation capabilities, however, the source code suggests that the activity sequence and durations can be modified after inferring a process model. However, it is unclear how this framework handles the simulation of case arrivals and resources.

SAMPLE (Grüger et al. 2022) uses *Data Petri Nets (DPN)* inferred from an event-log to generate a full simulation model. The strength of this approach is that the user can model any attribute observed in the data and represent it using the appropriate distribution, enabling the user to vary the distribution parameters of any datapoint represented by the DPN.

Using a radically different approach (Peeperkorn et al. 2022) train a Long Short-term Memory Recurrent Neural Network (LSTM-RNN) to represent the control flow of a process. However, this implementation is purely empirical and cannot be modified by the user to simulate different scenarios. Furthermore, time and resources were not included in this implemenation.

Identified research gap

The majority of the literature include functionality for modelling all four components of a business process simulation model: Case arrivals, activity sequences (control flow), duration distributions and resource availability. The control flow is mainly represented from a BPMN model, Petri Net or Process tree. Two approaches stand out in this aspect: process simulation using systems dynamics Pourbafrani and van der Aalst (2020) and Recurrent Neural Networks (Peeperkorn et al. 2022). The key advantage of approaches such as BPMN and Petri Nets for representing the control flow is the ability to represent activities executed in parallel, while keeping the number of parameters low, as these models are high-level abstractions compared to Markov chains (van der Aalst 2016).

On the other hand, the found frameworks based on these abstract process models do not represent temporal relationships multiple steps back in time, which is the key strength of Recurrent Neural Networks, often used in predictive process monitoring Evermann et al. (2016); Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa (2017); Navarin et al. (2018); Camargo et al. (2019a). Studying the performance of the memory-related aspects (as illustrated by Equation 4) of these models using Process trees, Petri Nets or

BPMN simulation models will thereby require modifications to the current found approaches.

As many of the proposed frameworks focus on the automation aspect of generating a business process simulation model, less emphasis is in many cases put on the details of the probabilistic distributions used to model the four process components: case arrivals, activity sequences, duration distributions and resource availability. This has the disadvantage that the resulting process simulation model becomes less transparent, and thereby less suited for testing hypotheses related to the data generating procedure (DGP).

Requirements for proposed framework

Based on the previous comparison of the existing frameworks, as well as the discussed use-cases in the introduction, a set of design requirements is defined below:

- **R1:** Follow theoretical DGP of a business process
  - Produce time-stamped synthetic event-logs
- **R2:** Possibility to specify distributions from a theoretical model
  - Case arrivals
  - Activity sequences and process complexity
  - Conditional duration distributions
  - Resource availability
- **R3:** Publicly available documentation
- **R4:** Open source
- **R5:** Implementation in Python programming language

R1 is introduced to highlight that the outcome of a simulation run should be a synthetic event-log that follow the general structure of an event-log originating from business process. **R2** firstly ensures that the data generating process is transparent and based on well-known distribution theory, as well that relevant aspects of the process can be varied and included as control variables in an experimental design. **R3** ensure that the methodology and assumptions behind the generated event-logs are transparent. **R4** ensures that the resulting framework is freely available and can be modified in any way a user see fit. **R5** ensures that the framework can be easily used in future research projects in the Predictive Process Monitoring literature.

Simulation approach

In the following, a proposed new simulation framework that addresses the requirements **R1-R5** listed in the introduction will be introduced. The framework is based on Python 3.8 and is dependent on the following packages: Numpy (Harris et al. 2020), Pandas (development team 2020) and Pomegranate (Schreiber 2018). The framework is dependent on 9 individual Algorithms to generate an event-log, which will be described in detail in the next sections. The framework uses Markov chains to generate the control flow (activity sequences), and uses the Hypoexponential distribution to simulate individual activity durations, where the spread of the average duration of individual activities at different time steps can be controlled. The time components include waiting times in the form of both stochastic and deterministic offsets. These components represent the uncertainty of resource availability, process stability and business hours. The framework generates an event-log mimicking the structure of a real world process, however, compared to existing simulation frameworks reviewed in the literature, the data generating process is purely theoretical and specified by the user.

The framework enables researchers to simulate processes with and without memory, which is implemented via first and higher-order Markov chains Ching et al. (2004). Furthermore, the process entropy (or complexity of the control flow) can be controlled such that a deterministic process (Minimum entropy), a completely random (Maximum entropy) or a user-specified (Medium entropy) process can be generated. The purpose of introducing these three levels is merely to enable the comparison between a perfectly predictable, completely unpredictable, and a user-defined scenario. The intended use of this framework is the testing of robustness in Predictive Process Monitoring, as discussed in the introduction.

As the framework is open source, any user with a need to do so may change the distributions used for either of the stochastic time components (e.g. Mixtures instead of Exponential and Hypoexponential distributions). The framework is purely based on parametric distributions, and the total trace throughput time can thereby be specified by a linear model as shown in Equation 25. As one of the key requirements of this framework is transparency, each component will be described in detail in the following.

Simulation components

A conceptual mapping between the data components and their related algorithms can be seen from Figure 1. At the highest level is the generated event-logs Θ_k, where each event-log is the result of a single simulation run. Each event-log consist of multiple traces $\Theta = \{Q_1, ..., Q_n\}$, which in turn consist of one or more activities $Q_i = \{e_1, ..., e_n\}$. Each activity has a total duration u_t (including waiting times),

which again can be split into multiple components as seen in the bottom of Figure 1.

The generation of traces can be split into the arrival times, which are modeled as a Poisson process (see section Trace arrivals). The activities and control-flow of the traces are modeled using Markov chains (memoryless process) and higher-order Markov chains (for processes with memory) which is described in section Transition matrices for the control flow, and Trace generation for trace generation. Finally, the activity offsets and trace duration are described in sections Activity offsets and Trace duration, respectively.

Trace arrivals

The timestamps of activities in an event log $\Theta = \{Q_1, ..., Q_n\}$ is modeled in continuous time, using an offset $T_0 \in \mathbb{R}$, assumed to be a number of arbitrary time units (days, months, years) after a fixed point in time: 01/01/1970 00:00:00, specified by the user. The beginning of the simulation period is denoted $t = 0$, indicating that no time has passed at this point. A trace Q_i is assumed to come from a *Poisson process*, such that the time between arrivals $x_i \in \mathbb{R}$ is drawn from an exponential distribution $Exp(\lambda)$. For the illustration of trace arrivals, the event-log notation earlier introduced is extended with a case-level attribute z, denoting the arrival time of the trace. The full procedure is demonstrated in Algorithm 1.

Algorithm 1: Generation of trace arrival times in a synthetic event-log.

Data: Time period upper boundary ψ, rate parameter g
Result: Event-log Θ
/* Placeholders: Event-log and arrival times, respectively. */
$\Theta \leftarrow \emptyset$
$\vec{z} \leftarrow \emptyset$
$i = 1$
/* While inside the simulation period $0 \leq t < \psi$ */
$t = 0$
while $t \leq \psi$ **do**
  /* Arrival time for trace i */
  Draw $x_i \sim Exp(\lambda = g)$
  /* Generate a new trace Q_i */
  $Q_{(c=i,z=x_i)}$
  /* Increase the timeline by the arrival time of Q_i */
  $t = t + x_i$
  /* Append Q_i to the event-log Θ */
  $\Theta \leftarrow Q_i$
  /* Append x_i to the arrival times vector $\vec{z}$ */
  $\vec{z} \leftarrow x_i$
  /* Increase case identifier */
  $i = i + 1$
end

Transition matrices

The transition matrices play a crucial role in the resulting process variants. For the used approach in this study, a vector of initial probabilities P^0 is firstly, which will be generated using Algorithm 2. For the subsequent transitions in a given trace, one or more transition matrices are needed, depending on the approach (memory versus memoryless process).

For a given transition matrix generated in a single experiment, each possible transition $P_{i,j}$ is defined in a manner that generates a specific type of control flow graph. In this framework three different types of transition matrices are possible: Minimum, medium and maximum entropy. The minimum and maximum settings are two extereme cases, where minimum represents a deterministic process, and maximum represents a random process with no order (absence of a process). For the medium entropy, the user can specify the number of transitions possible from each state, which in combination with the number of states can lead to different levels of complexity.

1. **Minimum entropy:** For each row P_i, only one state is possible
2. **Medium entropy:** For each row P_i, only two n are possible with probability distribution Ω
3. **Maximum entropy:** For each row P_i, all states are possible with equal probability

To generate each of these types of transition matrices, algorithms 3, 4 and 5 will describe the procedure in pseudo-code. The initial probabilities P^0 are described in the following.

Initial probabilities Depending on the approach used (memory or memoryless), multiple hypothetical transition probabilities needs to be generated. Common for both approaches is the need of the initial probabilities P^0. These can be generated as shown in Algorithm 2.

Minimum entropy The minimum entropy situation is where the simulated process is 100% conforming: it has no deviations from its intended control flow, and no rework. The minimum-entropy transition matrix generated in Algorithm 3 is the equivalent of a *multivariate hypergeometric distribution* for each step in the sequence. However, since only one outcome is possible (with probability one) per transition, the minimum entropy scenario is by definition a deterministic process.

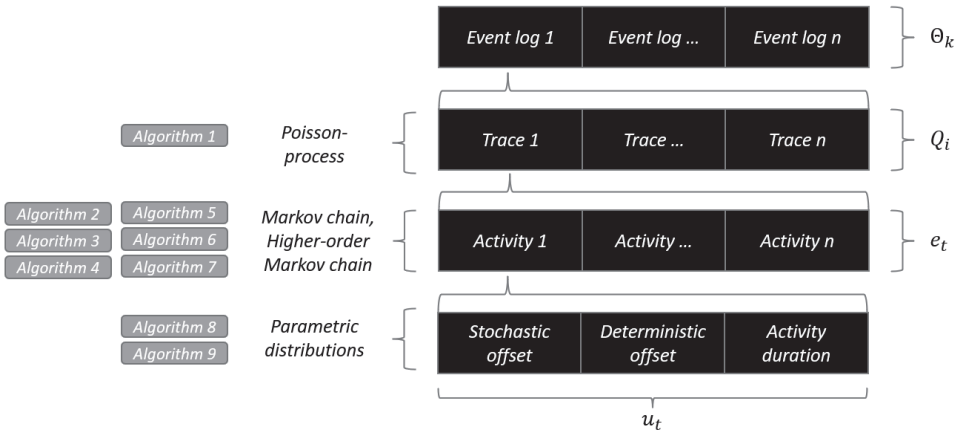


Figure 1. Overview of data components and related algorithms.

Algorithm 2: Generation of a initial probability vector P^0

Data: A set of states $D = \{S_1, ..., S_{|D|}\}$, probability distribution Ω
Result: Initial probabilities P^0
/* Placeholder */
$P^0 \leftarrow \emptyset$
/* For every state/element in D, draw a probability and append to P^0 */
for d *in* D **do**
  Draw x_d from distribution Ω
  $P^0 \leftarrow x_d$
end
/* Scale the values in P^0, to reflect that they are probabilities of mutually-exclusive states */
$S_{sum} = \sum_{i=1}^{N}(P_i^0)$
$P^0 = \frac{1}{S_{sum}} P^0$

The output of Algorithm 3 is a transition matrix P, where only one transition is possible between each state until reaching the absorption state. In this particular case, the trace length can only be controlled via the number of states in D.

Maximum entropy The maximum entropy scenario represents a theoretical situation where a process is completely random and thereby unpredictable. This scenario is only included as a contrast to the space of possible processes that can be generated by the medium entropy setting (presented in the next section). The transition matrix generated in Algorithm 4 will have equal probabilities for each of the possible states in D, for all transitions. Each transition is

Algorithm 3: Generation of transition matrix: Minimum entropy

Data: A set of states $D = \{S_1, ..., S_{|D|}\}$
Result: A transition matrix P
$P \leftarrow \emptyset$
/* For every sate in D, generate a new row-vector R */
for d *in* D **do**
  /* Initialize R with zeros */
  $R \leftarrow \vec{0}$
  /* For the last state, only absorption is possible */
  if $d = |D|$ **then**
    /* Set the last index of R to one */
    $R_{(d=|R|)} = 1$
  else
    /* Generate transitions: Randomly select a state from D */
    $s = SelectByRandom(D)$
    while $s = d$ **do**
      /* If the to and from state is the same, draw again */
      $s = SelectByRandom(D)$
    end
    /* Remove the selected state from D */
    $D = s \setminus D$
    /* Set the *s'th* index to probability one */
    $R_{(d=s)} = 1$
  end
  /* Append the row vector into P */
  $P \leftarrow R$
end

thus equivalent to a multinomial distribution with identical weights of each class.

Algorithm 4: Generation of transition matrix: Maximum entropy

Data: A set of states $D = \{S_1, ..., S_{|D|}\}$
Result: A transition matrix P
/* Placeholder for transition matrix P */
$P \leftarrow \emptyset$
/* For every sate in D, generate a new row-vector R */
for d *in* D **do**
  /* Every event is equally likely, so R is a vector of ones */
  $R \leftarrow \vec{1}$
  /* Normalize into probability space */
  $R = \frac{R}{\sum R}$
  /* Append the row vector into P */
  $P \leftarrow R$
end

Medium entropy The medium entropy scenario can be adapted to represent different levels of complexity, by changing the number of states D, and number of possible transitions n. The transition matrix P^k generated from Algorithm 5, model a process that has transitions generated from a multinomial distribution, where n is the number of possible categories. A while-loop in the last section of Algorithm 5 avoids *livelock* (never-ending sequences) by repeatedly adding transitions to the absorption state, if the sum of the absorption probabilities across all states is not equal or greater than the threshold κ.

Algorithm 5: Generation of transition matrix: Medium entropy

Data: A set of states $D = \{S_1, ..., S_{|D|}\}$, probability distribution Ω, number of non-zero state transitions per row n, probability threshold for transitions to absorption κ
Result: Transition matrix P
/* Placeholder */
$P \leftarrow \emptyset$
/* For every state/element in D */
for d *in* D **do**
  /* Initiate L as a zero-vector of length $|D|$ */
  $L \leftarrow \vec{0}$
  /* Draw n states from D, without replacement */
  $S \leftarrow$ Draw $\{s_i, ..., s_n\}$ from $U(1, |D|) \in \mathbb{N}$
  /* Generate probabilities and replace zeros at each index $L_{i=s}$ */
  for s *in* S **do**
    Draw $x_s \sim \Omega \in \mathbb{R}$
    $L_{i=s} \leftarrow x_s$
  end
  /* Normalize L, and append to P */
  $L = \frac{L}{\sum L}$
  $\mathbf{P} \leftarrow L$
  /* If the sum of probabilities of transitions to absorption is less than κ, add more transitions */
  while $\sum P[0 : |D|, |D|] < \kappa$ **do**
    /* Draw a random state, which is not the absorption state. */
    $d \leftarrow$ Draw s from $U(1, |D| - 1) \in \mathbb{N}$
    /* Add random probability to absorption */
    $P[d, |D|] \leftarrow$ Draw $x_s \sim \Omega \in \mathbb{R}$
    /* Normalize row d to probability-space */
    $P[d, :] = \frac{P[d,:]}{\sum P[d,:]}$
  end
end

Trace generation

Generating the traces in a simulation experiment consist of two steps: Firstly, the control-flow is generated using the transition matrices described in section Transition matrices. Next, the transition matrices are used for probabilistic sampling using either of the process memory approaches. The procedure of the two approaches will be described in more detail in the following.

Memoryless process In cases where the process is assumed to be memoryless, a first-order absorbing Markov chain ($k = 1$) is used to model the transitions between states (activities a) in the process until termination. This is represented by a $|D| \times |D|$ matrix of transition probabilities P, which is treated as a conditional multinomial distribution. Sampling a trace from P will continue until the absorbing state $D_{|D|}$ is met. The trace length is determined by the number of states in D, as well as the assumptions of the control flow (Algorithm 3, 4 or 5). The full procedure can be seen in Algorithm 6.

Process with memory For the implementation in Algorithm 7, a set of initial probabilities and subsequent transition matrices is used to generate each state, until the stopping criteria is met (reaching the absorption state). The implementation does thereby not use a single k*'th* order tensor as illustrated in Equation 4, but an initial probability

Algorithm 6: Trace generation for a memoryless process

Data: Set of possible states $D = \{S_1, ..., S_{|D|}\}$ (activities), Algorithm 2, Algorithm 3, Algorithm 4, Algorithm 5
Result: Trace $Q = \{e_{(1,a)}, ..., e_{(t,a)}\}$

```
/* Initialize                                   */
```
$Q \leftarrow \emptyset$
$t \leftarrow 1$
```
/* Generate probability
   distributions P^0, P                         */
```
$P^0 \leftarrow Algorithm2(D)$
$P \leftarrow Algorithm3(D, ..)$, $Algorithm4(D, ..)$ or $Algorithm5(D, ..)$
```
/* Get initial activity and append
   */
```
$Q \leftarrow$ draw $a_{t=1}$ from P^0
```
/* While the absorption state has
   not yet been reached, do                     */
```
while $a_t \neq D_{|D|}$ **do**
    $t = t + 1$
    `/* Sample from first-order Markov chain */`
    $a_t \leftarrow$ Draw from distribution corresponding to the $a_{(t)}$-th row of P
    `/* Append activity to trace */`
    $Q \leftarrow a_t$
end

Algorithm 7: Trace generation for a process with memory

Data: Set of possible states $D = \{S_1, \ldots, S_{|D|}\}$, Markov chain of order k, Algorithm 2, Algorithm 4, Algorithm 5
Result: Trace $Q = \{e_{(1,a)}, \ldots, e_{(t,a)}\}$

```
/* Initialize                                   */
```
$\Phi \leftarrow \emptyset$
$Q \leftarrow \emptyset$
$t = 0$
```
/* Generate initial distribution P^0
   */
```
$P^0 \leftarrow Algorithm2(D)$
```
/* Generate probability
   distributions (P^1, P^2, ..., P^k)           */
```
for i *in* $\{1, \ldots, k\}$ **do**
    `/* Generate the conditional transition probabilities for D */`
    $P^i \leftarrow Algorithm5(D, \ldots)$ or $Algorithm4(D, \ldots)$
    if $i > 1$ **then**
        `/* Generate the expanded state space D^i */`
        $D^i \leftarrow CProduct(D, i)$
        `/* Generate the conditional transition probabilities for D^i */`
        $P^i \leftarrow Algorithm5(D^i, \ldots)$ or $Algorithm4(D^i, \ldots)$
    `/* Append each transition matrix to ρ */`
    $\Phi \leftarrow P^i$
end
```
/* Sampling: Get initial activity */
```
$Q \leftarrow$ draw $a_{t=0}$ from P^0
```
/* While the absorption state has
   not yet been reached, do                     */
```
while $a_t \neq D_{|D|}$ **do**
    $t = t + 1$
    if $t < k$ **then**
        `/* Sample from the t'th transition matrix in Φ */`
        $a_t \leftarrow$ Draw from distribution corresponding to the Q*'th* row of transition matrix Φ_t
    else
        `/* Sample from P^k */`
        $a_t \leftarrow$ Draw from distribution corresponding to the $hd^k(Q)$*'th* row of transition matrix P^k
    end
    `/* Append activity to the trace */`
    $Q \leftarrow a_t$
end

vector $\vec{P^0}$ and transition matrices $\Phi = \{P^0, P^1, ..., P^k\}$ of increasing size.

Depending on the amount of memory k, the size of each transition matrix $P^i \in \{1, ..., k\}$ exponentially increases, as each state d in the new state space D^i depends on the pattern of states the last i time steps. To generate the new expanded state space D^i, a function $CProduct(D, i)$ is defined, which produces the Cartesian product of D, i times: $D^{(i=3)} = D \times D \times D$. In practical terms, this results in a sate space of all possible combinations of a sequence of length 3 (representing the path of the last 3 steps).

To generate a trace Q, the sequence until the k*'th* time step is firstly generated, and if the absorption state is not yet reached, a further sequence of length k is generated, using the conditional probability P^k of the last k generated steps in $hd^k(Q)$, where hd^k is the *head* operator retrieving the last k items in Q. This process continues until the absorbing state $D_{|D|}$ is reached. The Python-implementation uses the $MarkovChain$ object from the Pomegranate Python library (Schreiber 2018) for Algorithm 7.

Sequence encoding A generated example sequence Q_i of length $|Q| = 3$, can be represented in multiple ways: a character-vector of length $|Q|$, or encoded as a binary vector of size $|Q| \times |D|$, with ones along the $|D|$-axis, when the d*'th* state is observed in Q_t for $t = \{1, \ldots, |Q|\}$. For mathematical convenience, the binary encoding is used in this framework. A generated sequence/trace i will thus be referred to as $V_{(i,T,D)}$, where i denote the trace identifier in

the event-log Θ, $T = (|Q|)$ the discrete time dimension, and D is the state-space including the absorbing state S_{END}. For the example sequence $Q_i = \{S_2, S_1, S_{END}\}$, from the state-space $D = \{S_1, S_2, S_{END}\}$, the first time step will at time step $t = 1$ result in the following binary vector: $V_{(i,t=1,D)} = (S_1, S_2, S_{END}) = (0, 1, 0)$.

Activity durations

All activity durations $y_t \in \mathbb{R}$ are drawn from the exponential distribution, which uses a single *rate* parameter λ to specify the probability density function: $f(x, \lambda) = \lambda e^{-\lambda x}$.

Each possible timestep-state combination has its own rate parameter $\lambda_{(d,t)}$ which can be drawn from any arbitrary continuous distribution Ω. In the implementation of this framework the uniform distribution is used, $\Omega \sim Uniform(0.0001, \xi)$, where ξ is the upper limit.

For a full trace, this effectively becomes $|D|$ Hypoexponential distributions: $\sum_{t=1}^{|Q|} u_{(d,t)} \sim Hypo(\Lambda_d \sim \Omega(\xi))$, where Λ_d is a $|D| \times |Q|$ matrix containing the rate parameters. The isolated activity duration of the event d at time step t can thus be expressed as:

$$v_{(d,t)} = Exp(\Lambda_{(d,t)}) \tag{18}$$

Activity offsets

The time between the activity of an event e_1 is completed, and the next activity of e_2 can begin will be referred to as the activity *offset*. In other words, this is the delay before processing of a given activity will begin. The offsets represented in this framework consist of both stochastic and deterministic components. The respective offsets are assumed to be a function of the following factors:

- **Stochastic:**
  - Resource availability
  - Stability of the process
- **Deterministic:**
  - Outside business hours

Activities are assumed to be processed via first come first served (FCFS) priority, where the waiting time is simulated through stochastic resource availability (section Resource availability). Once a resource is ready to process the activity, a process stability offset can be be included (section Process stability), and the deterministic offset will then be calculated via the procedure in section Business hours (based on the stochastic offsets). Afterwards, the activity is processed, and its duration is defined in Equation 18. An overview of the offsets is shown in Figure 2.

Resource availability As most business processes involve a resource or agent to process the activity, resources are also included in the parametric simulation model. To avoid a multi-level type simulation model, cases are assumed to be identically independently distributed, in that resource availability is modeled using the binomial distribution. In short, every activity will have an *activity resource offset* value h, which represents the waiting time for acquiring an idle resource. Algorithm 8 provides an overview of the generation of the *activity resource offset*.

Algorithm 8: Generate resource availability offset for a single activity

Data: Number of agents n, Probability of idle agent p, Time between requests m
Result: Agent availability offset h
`/* Agent availability offset starts at 0 */`
$h = 0$
`/* Keep trying until resource is available */`
while $k < 1$ **do**
  `/* Get the number of trials before success */`
  Draw k from $Binom(n, p)$
  `/* If k < 1, add a waiting time penalty */`
  if $k < 1$ **then**
    $h = h + m$
end

Process stability Depending on the assumptions of the process, stochastic offsets can be added to represent process (in)stability. This enables the ability to generate extreme observations, which can be present at any event e_t within a trace Q. The process stability offset b can be represented using any continuous distribution, however, it is implemented as the exponential distribution in this framework:

$$b \sim Exp(\lambda) \tag{19}$$

Consequently, the combined stochastic offsets of a process activity d at time step t can be formulated as:

$$m_{(d,t)} = h_{(d,t)} + b_{(d,t)} \tag{20}$$

Business hours To calculate the deterministic offsets related to business hours (open versus closed), the time point during the week where the activity was scheduled to be performed needs to be identified. This is achieved using Equation 21:

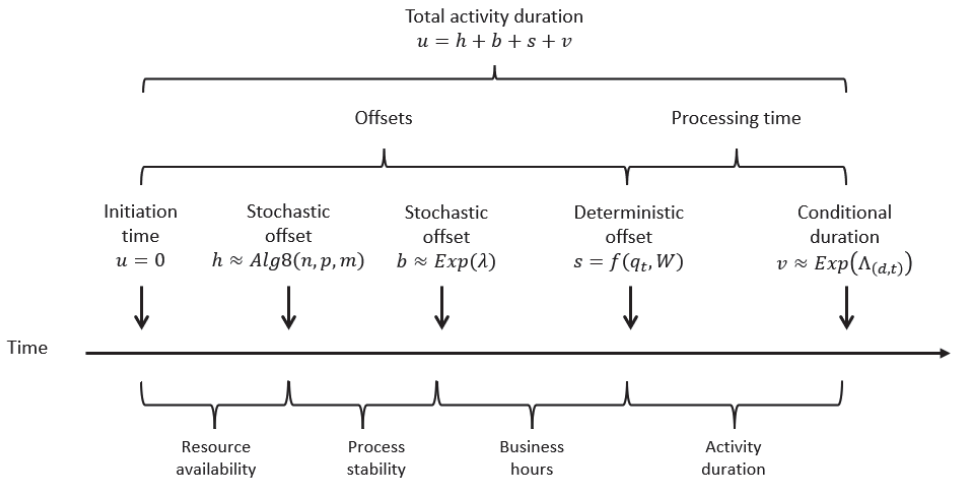


Figure 2. Overview of activity offsets. Exp refer to the exponential distribution.

$$q_t = ((Q_{(z)} + [t > 1](\sum_{j=1}^{t-1} u_j)) + m_{(d,t)}) \mod \eta \quad (21)$$

Where q_t denote the scheduled start time since the beginning of the week, $Q_{(z)}$ is the arrival time of the *i'th* trace $Q_i \in \Theta$ (see Algorithm 1). The term $[t > 1](\sum_{j=1}^{t-1} u_j)$, is the total duration of all previous events of the *i'th* trace, given that there are any. The expression: $[t > 1]$ will assume the value of one when true, and zero otherwise (Iverson 1962). $m_{(d,t)}$ is the sum of the stochastic offsets of the current event e_t (equation 20), and finally, η is the number of time units within a week (days, hours, minutes).

To get the offsets for work scheduled outside business hours, the time until work can continue is added using the function f in Equation 22. The function returns a conditional value based on logical constraints and the value of q_t.

$$f(q_t, W) = [q_t \geq W_{(1,1)} < W_{(1,2)}](W_{(1,2)} - q_t) \quad (22)$$

Here the values $(W_{(1,1)}, W_{(1,2)})$ denote the respective start and end of the first interval of the week, wherein the activity e_t cannot begin due to business rules (process is *offline*), and will thus be given an offset. Using this logic, any amount of intervals can be introduced when W is a $n \times 2$ matrix, where n is the number of intervals the business is process is offline:

$$W = \begin{bmatrix} W_{(1,1)} & W_{(1,2)} \\ \vdots & \vdots \\ W_{(i,1)} & W_{(i,2)} \\ \vdots & \vdots \\ W_{(n,1)} & W_{(n,2)} \end{bmatrix}$$

To include all intervals, Equation 22 can be reformulated as:

$$f(q_t, W) = \sum_{i=1}^{n} [q_t \geq W_{(i,1)} < W_{(i,2)}](W_{(i,2)} - q_t) \quad (23)$$

Finally, the total offset can be calculated as:

$$O_{(d,t)} = f((Q_{(z)} + [t > 1] \sum_{j=1}^{t-1} v_{(d,t=j)} + H_{(d,t)} + B_{(d,t)}) \mod \eta, W)_{(d,t)} \quad (24)$$

Where the matrix $H_{(d,t)}$ denote the resource availability offsets across all time steps in the trace $t \in \{1, ..., |Q_i|\}$ and activities $d \in D$. Similarly, $B_{(d,t)}$ contain the process stability offsets, and $O_{(d,t)}$ represent the total offsets.

Time components

In the previous sections the individual time components (duration and offsets) have been introduced separately. Algorithm 9 demonstrates how the individual components are generated in order to form the timestamp of each event

n_t, as well as the duration including the offsets u_t. The total duration of an event e_t is formulated as U_t.

Algorithm 9: Generation of trace time components

Data: A matrix of offline-intervals W, the number of time units within a week η, a complete trace with a sequence of events $Q = \{e_1, ..., e_{|Q|}\}$, Resource availability parameters n, p, m, process stability rate r, a matrix of rate parameters $\Lambda_{d,t}$

Result: Time components N_t

/* Placeholders */

$N \leftarrow \emptyset$

/* For each timestep in the current trace */

for t *in* $\{1, ..., |Q|\}$ **do**

/* Resource availability offset */

$h_t \sim Algorithm8(n, p, m)$

/* Process stability offset */

$b_t \sim Exp(\lambda = r)$

/* Position during the work-week (time since monday) */

$q_t = (([t > 1](\sum_{j=1}^{t-1} u_j)) + b_t + h_t) \mod \eta$

/* Business hours offset */

$s_t = f(q_t, W)$

/* Get index d for current timestep t, to get individual rate parameter */

$d = Q_{(t=t,d)}$

$v_t \sim Exp(\Lambda_{(d=d,t=t)})$

/* Total duration of the activity */

$u_t = h_t + b_t + s_t + v_t$

/* Generate timestamps of activity starttime n_t */

if $t = 1$ **then**

$n_t = Q_{(z)} + h_t + b_t + s_t$

else

$n_t = n_t + u_t$

end

/* Append components to N */

$N_t \leftarrow (u_t, h_t, b_t, s_t, v_t, n_t)$

end

Trace duration

The simulation Equation for the total trace duration (throughput time) y_i of a trace Q_i has the form:

$$y_i = \sum_{d=1}^{D} \sum_{t=1}^{T} G_{(i,d,t)} E_{(i,d,t)} + O_{(i,d,t)} E_{(i,d,t)} \tag{25}$$

$$G_{(i,d,t)} \sim Hypo(\Lambda_{(d,t)}) \tag{26}$$

Where $\Lambda_{(d,t)}$ is a matrix of size $|D| \times T$ containing the individual rate parameters for each state-time pair, $O_{(i,d,t)}$ refer to the combined offset of each activity, and $E_{(i,d,t)}$ refer to a one-hot encoded vector of the state transitions over the full trace Q_i (see section Sequence encoding).

Equivalently, the durations for all individual activities $e_t \in Q_i$ can be expressed as:

$$u_{(i,d,t)} = G_{(i,d,t)} E_{(i,d,t)} + O_{(i,d,t)} E_{(i,d,t)} \tag{27}$$

Simulation parameters

Table 3 provides an overview of the input parameters of the simulation framework.

Simulation test

In the following, a set of simulation runs will be presented in order to demonstrate the capabilities of the framework. To achieve this, a full factorial design over the factors and levels in Table 4 was generated with 10 replications. This resulted in a total of 960 generated event-logs of variable size. Some levels do not influence each other (*process_memory* and *process_type=memoryless*), but were kept in the design table to ensure a balanced design. The simulation framework will be evaluated in terms of the CPU-time used, as well as the trace distributions, duration distributions. A more qualitative evaluation will be performed to illustrate examples of the resulting control flows for each of the levels of entropy, as well as the structure of a single eventlog generated with this framework.

Simulation performance: CPU time

The simulation test was conducted in *python 3.8*, on a workstation with a Intel i9-13900KF CPU and 64 GB DDR5 RAM. The code in its current form has not been optimized for multi-core processing, which leads to significantly longer runtime for the process with memory.

In Figure 3 the distribution of time in seconds to generate an event-log with and without memory versus different levels of entropy can be seen. The time to generate the event-logs from a Higher-order Markov chain is orders of magnitude larger than the memoryless process. This is due to the increased size in the number of parameters, as well as the fact that this framework do not leverage multiple cores when generating sequences from a Markov chain. The relative difference in runtime between *medium* and *maximum* entropy, is due to the fewer possible paths in the *medium* variant, which is more likely to lead to longer traces, as the transition might get temporarily ”stuck” between states before reaching the absorption state. The minimum entropy

Parameter	Explanation
n_traces	Number of traces to generate.
d_size	The size of the state-space of the process (number of activities).
process_entropy	The desired level of entropy in the generated process. This can be set to either *minimal*, *medium*, or *maximal*.
process_type	This specifies whether the data generating process is memoryless or have k orders of memory.
process_memory k	The order of the process with memory.
n_transitions	Also unique to the *medium*-level entropy process, this parameter specify the number of possible transitions from each state in D.
inter_arrival_time	The parameter specifying the time between arrivals in Algorithm 1.
process_stability_scale	The parameter specifying average offsets due to lack of process stability (equation 19).
resource_availability_p	The probability p that an agent is available at any time t (algorithm 8).
resource_availability_n	The number of resources allocated to the process, at any time t (algorithm 8).
resource_availability_m	The amount of time between requests when no resource is currently available (algorithm 8).
activity_duration_lambda_range	The range of the parameters specifying average duration for each $D \times T$ combination in $\Lambda_{(d,t)}$ (equation 18). Values are drawn from the uniform distribution.
deterministic_offset	The type of deterministic offsets related to business hours. This can be specified as *weekdays* with open hours from 6-18, Monday to Friday, or alternatively 6-18 on all weekdays. The business hours intervals can be changed by the user from the matrix W introduced with Equation 22.

Table 3. *Input parameters for the simulation framework*

Factor	Levels
n_traces	500
d_size	5, 10
process_entropy	*Minimal*, *medium*, *maximal*
process_type	Memory, memory-less
process_memory	2, 4
n_transitions	3, 5
inter_arrival_time	1.5
process_stability_scale	0.1
resource_availability_p	0.5
resource_availability_n	3
resource_availability_m	0.041 (15 minutes)
activity_duration_lambda_range	1, 5
deterministic_offset	Weekdays

Table 4. *Settings for the simulation test*

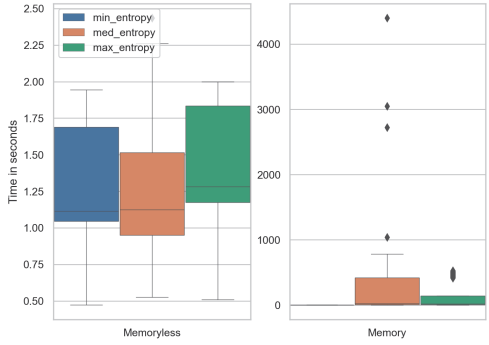


Figure 3. Simulation performance: CPU time in seconds.

setting for the process with memory is the fastest, as it is a deterministic process.

Trace distributions

The distribution of the average and maximal trace lengths of each simulated event-log can be seen from Figure 4. As seen from the fliers (black dots) of the two box plots, the

$med\_entropy$ settings are most likely to generate outliers in terms of trace length. As previously discussed, these transition matrices are restricted in the number of possible paths the process can take (defined by the number of transitions in Algorithm 5), these processes are thereby more likely to get temporarily "stuck" before reaching the absorbing state. As Figure 4 shows, this behaviour becomes even more extreme when the process has memory.

Event distributions

For the simulation runs using a *memoryless* process, Figure 5 illustrates the distribution of events within the first 25 time steps. The box plots represent the variation in frequency for each event e_t across the simulated event-logs, separated by the level of entropy. Each individual chart represent a given size of the state space (3, 5), combined with a given number of transitions (5, 10). The individual plots both illustrates frequencies of events, but trace lengths can also be inferred from it.

Focusing on the settings with $min\_entropy$ (blue), there is no variation in the frequency at each time step (see black median lines at y = 500), and the box plot simply becomes a thin static line. Furthermore, the maximal number of events observed is equal to the size of the state space. As this is a deterministic process with only one trace variant, this is expected.

For the event-logs simulated with $med\_entropy$ (orange) a high level of variation in the frequency can be seen in each of the four charts, with the largest variation being in settings with a state space of 10. These experiments also have the highest average frequency of events at every point in time, which also means longer traces. This is expected, as the number of transitions, and the size of the state space has an influence on the trace length, and thereby the overall distribution of events/time steps observed.

For the settings with $max\_entropy$ (green), there are far less variation compared to $med\_entropy$. The explanation here lies in the transition matrices generated for the *medium* entropy process in Algorithm 5, as this generates unique, restricted transition matrices fulfilling the specified number of transitions and size of state space. Across multiple runs with the same setting the transition matrices will be different, and thereby represent different processes (and thereby greater variation).

The settings with *maximum* entropy (see Algorithm 4), will on the other hand produce the same transition matrix across multiple runs with the same setting. All possible trace variants will thereby be represented given enough samples, which leads to a more identical distribution across multiple runs with the same setting.

Moving on to Figure 6 which includes only distributions for runs with memory, a much larger degree of variation can be seen for the setting with $med\_entropy$. The explanation here is similar to the *memoryless* process, as the logic behind the transition matrices are the same. The major difference here is the size of the transition matrices, where the *memoryless* process is defined by an initial probability vector P^0 and a transition matrix P. In contrast, process with memory rely on P^0 and a set of k transition matrices of increasing size. For the setting with $min\_entropy$, each transition matrix $P^i \in \Phi$ will be unique. Comparing the results of $min\_entropy$ and $max\_entropy$, similar results as for the *memoryless* process are found.

Duration distributions

Looking at the duration distribution in Figure 7, it can be seen that the variation in the average duration of the activities change as the $activity\_duration\_lambda\_range$ parameter is altered (shown with colors blue and orange). This parameter specifies the upper boundary of the uniform distribution used to draw the individual rate parameters specifying each activity duration distribution at each possible time step $\Lambda_{(d,t)}$. As expected, an upper range of 5 (which means $Uniform(0, 5)$), leads to a higher observed median activity duration, and a higher spread in the distribution.

For the average activity durations shown in Figure 8, the mean (represented by the colored lines) fluctuates in a increasing manner as the trace becomes longer. As only 14 runs had traces with more than 100 events, the fluctuating behavior of the mean after this point is purely due to a low number of event-logs with this many events.

Trace example

An example trace is illustrated in tables 5 and 6. Table 5 show the simulated values for each of the continuous time components, which are included in each of the generated event-logs. For reference, see Figure 2 for an overview of the definition of time components and variable names. In Table 6, UNIX-timestamps are generated with a starting date of 2023-01-01 00:00. The $arrival\_datetime$ denote the point in time where the trace is generated, and $start\_datetime$ denote when a resource started working on the activity, finally, $end\_datetime$ denotes the point in time when the activity is completed. As seen by this example, activity f ends Saturday midnight, and as new activities cannot start

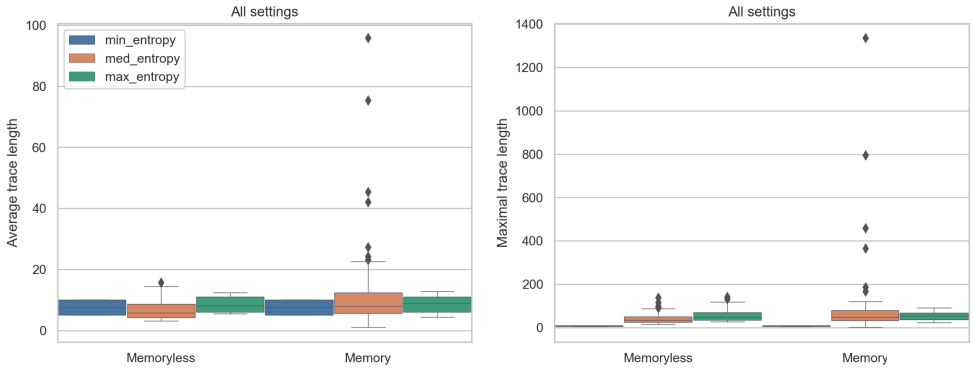


Figure 4. Distribution of trace lengths in each generated event-log.

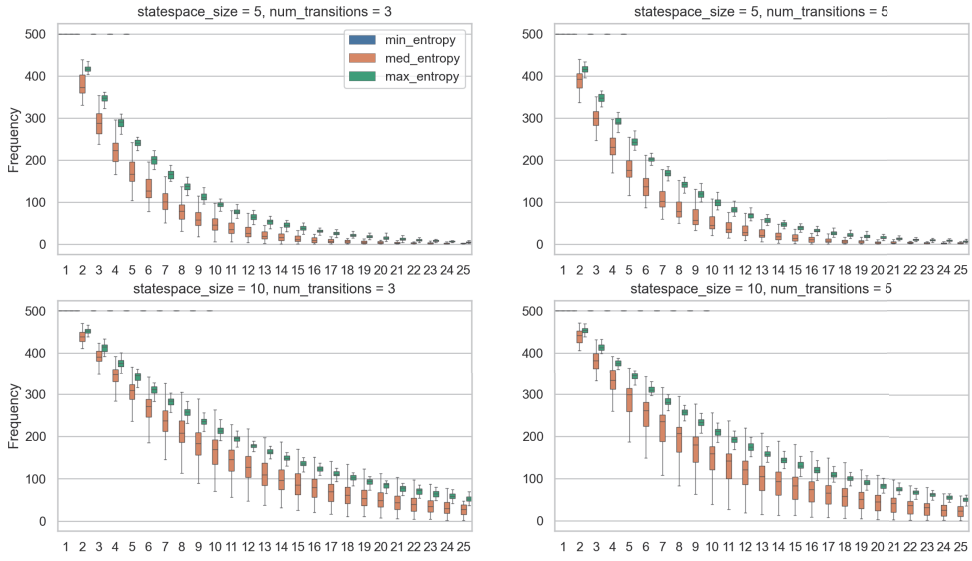


Figure 5. Memoryless process: Distribution of events 1 to 25, grouped by process entropy (color).

during weekends, activity e starts Monday morning at 06:00 instead.

Control-flow examples

To demonstrate the resulting control flows of the event-logs generated by the simulation framework, three examples of BPMN models have been created from the event-logs using Inductive miner (van der Aalst 2016) and the PM4PY Python library (Berti et al. 2019). As the simulation test consist of a total of 2304 unique event-logs, the inspection of the process diagrams has to be qualitative. The examples thereby show BPMN models of event-logs generated with *minimum*, *maxium* and *medium* entropy using the *memoryless* process and a statespace of size 5.

As seen by Figure 9, there is only one possible path in the control-flow of the minimum-entropy process, as it is defined as a deterministic process in the activity sequence (Algorithm 3). As previously mentioned, this is to simulate an example of the simplest possible process with no variation or uncertainty in it, which can be useful as a contrast to different levels of complexity.

The maximum entropy example in Figure 10, illustrates a process where all possible combinations of transitions within a given time-span is possible with identical probability (as defined in Algorithm 4). As every possible transition has identical probability, this represents a theoretical scenario that is completely random and thereby impossible to predict from a Machine learning perspective.

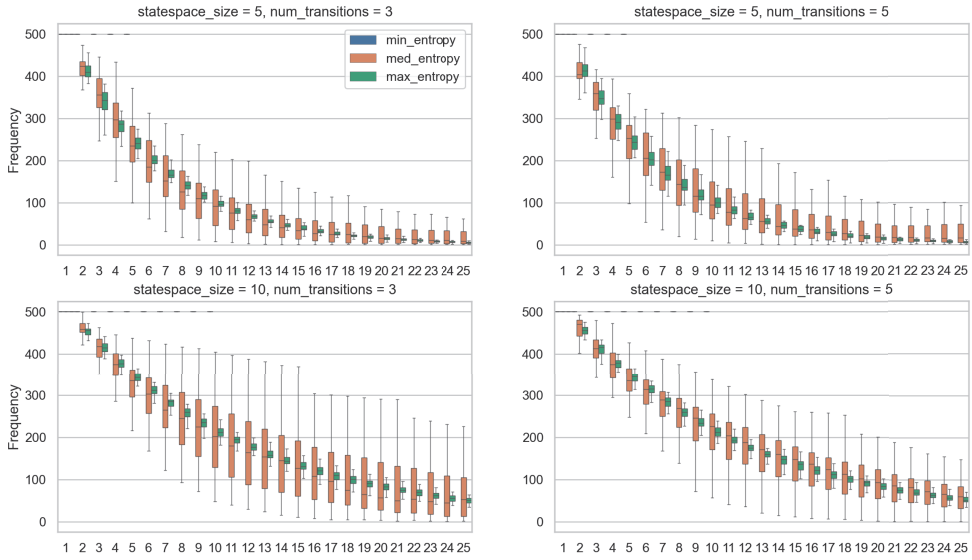


Figure 6. Memory process: Distribution of time steps from 1 to 25, grouped by process entropy (color).

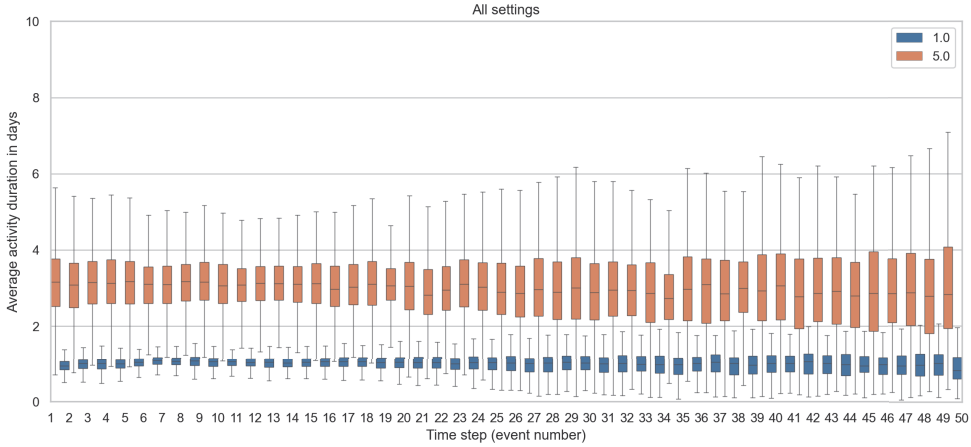


Figure 7. Average duration: Time steps from 0 to 50. The color denote the upper range of the $activity\_duration\_lambda\_range$ parameter.

caseid	activity	t	z_t	n_t	q_t	h_t	b_t	s_t	v_t	u_t
2	f	1	4.78	4.78	4.87	0.041	0.05	0.00	0.51	0.60
2	e	2	4.78	5.38	5.43	0.041	0.01	2.07	0.11	2.23
2	c	3	4.78	7.61	0.77	0.082	0.08	0.00	0.87	1.04

Table 5. Example-trace part one: Overview of simulated attributes

caseid	activity	t	arrival_datetime	start_datetime	end_datetime	start_day
2	f	1	2023-01-06 12:38:18	2023-01-06 14:51:18	2023-01-07 03:07:40	Friday
2	e	2	2023-01-06 12:38:18	2023-01-09 06:00:00	2023-01-09 08:44:23	Monday
2	c	3	2023-01-06 12:38:18	2023-01-09 12:35:46	2023-01-10 09:35:42	Monday

Table 6. Example-trace part two: Overview of simulated attributes

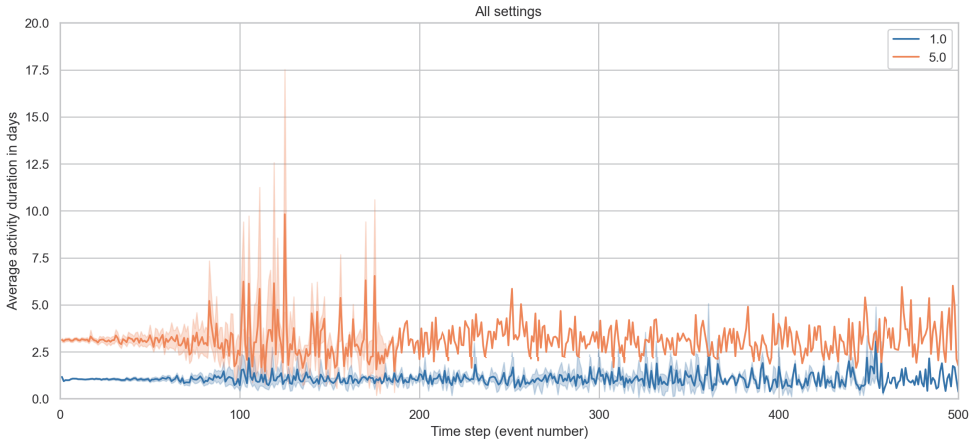


Figure 8. Average duration: Time steps from 0 to 500. The line color denote the upper range of the $activity\_duration\_lambda\_range$ parameter.

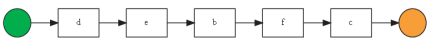


Figure 9. Minimum entropy BPMN: Memoryless process

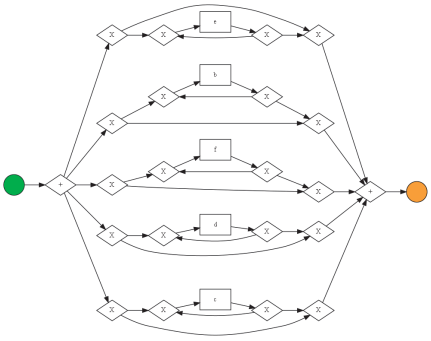


Figure 10. Maximum entropy BPMN: Memoryless process

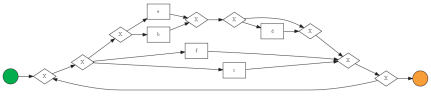


Figure 11. Medium entropy BPMN: Memoryless process

Finally, the *medium* entropy setting in Figure 11 demonstrates a process between the two extremes of *minimum* and *maximum* entropy, which is controlled by the combination of $n\_transitions$ and d_size (state space) parameters. Using these two parameters, this process-type can more closely resemble a process that is likely to be observed in a real business process, compared to *minimum* and *maximum* entropy. From Figure 11 a *memoryless* process with 3 transitions and a state space of 5 possible activities can be observed.

Discussion and conclusion

Aiming at contributing to the research in the field of Predictive process monitoring (Evermann et al. 2016; Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017; Navarin et al. 2018; Teinemaa et al. 2018; Verenich et al. 2019), and more specifically the robustness assessment of existing and future proposals, this paper has introduced a new framework for the generation of synthetic event-log data. Compared to existing open source approaches in the field, this framework is purely based on a theoretical data generating process. Compared to the existing frameworks, this framework has the disadvantage that it can not be calibrated to existing event-logs (in its current form).

On the other hand, the advantage of the framework in the form it is presented in this paper, is that the data generating procedure is transparent and based on well-known statistical distributions. The influence of data-related factors such as process complexity and temporal variability in the duration distribution can now easily be assessed. As demonstrated in the previous sections, these settings can be controlled and systematically changed in an experimental design.

Compared to the existing solutions based on BPMN models (Camargo et al. 2019b; López-Pintado and Dumas 2022; Fracca et al. 2021), Petri Nets (Pourbafrani et al. 2021b; Pegoraro et al. 2021; Grüger et al. 2022) and Process Trees (Pourbafrani et al. 2021a) found in the literature review, this framework uses Markov chains to represent the control flow of a simulation model. This approach has the disadvantage that it does not support concurrent/parallel processing of events, unless a significantly higher number of states is introduced (with multiple activities processed in

parallel encoded as separate states (van der Aalst 2016)). However, using Markov chains to represent the control flow, the process can be represented without abstractions, at the cost of the number of parameters. A higher-order Markov chain can thereby represent a process with no abstractions, where any event e_t in the trace is dependent on the complete history within the trace Q_i. As previously mentioned, this enables the simulation of unique temporal patterns in the duration distributions of activities, as well as activity sequences. This is especially relevant for the study on Long Short-term Memory Recurrent Neural Networks (Hochreiter and Schmidhuber 1997), which are known to perform well in prediction problems with long input sequences such as event-log data (Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa 2017; Navarin et al. 2018). This framework can therefore serve as an additional benchmark that enables a deeper understanding of data-related factors, and their influence on the performance of particular approaches in predictive process monitoring.

However, as this framework was developed for model robustness assessment in Predictive process monitoring, it has limitations that makes it less suitable for other branches of research. For instance, the individual behavior of resources or interruptions cannot be analyzed from this framework, as they are assumed to be represented by distributions. This is for instance a limitation for research in prescriptive process monitoring, where a calibrated agent-based simulation model such as (López-Pintado and Dumas 2022) would be more appropriate. Furthermore, the initial version of this framework also only supports the use of Exponential and Hypoexponential distributions due to their convenient relationship that enables the trace duration formulation in Equation 25.

In future iterations, functionality such as conditional transition matrices and duration distributions based on simulated case attributes can be introduced. Another possibility is to add more complexity to the duration distributions via time-dependent rate-parameters for the activity duration distributions $\Lambda_{(d,t)}$. Mixture distributions is also a possible direction to achieve this. Currently, the proposed framework is not optimized for multi-core processing, which could decrease the time it takes to generate multiple event-logs. However, this was not seen as a key design priority in this first version of this framework. The source code is publicly available from: https://github.com/Mikeriess/SBPS_framework, where practical instructions and examples are available. Readers interested in participating in the future development of the framework are welcome to submit pull requests or contact the author.

References

Aalst W (2015) Business process simulation survival guide. *Handbook on Business Process Management 1. International Handbooks on Information Systems.* : 337–370DOI:10.1007/978-3-642-45100-3_15.

Altiok T and Melamed B (2007) Chapter 2 - discrete event simulation. In: Altiok T and Melamed B (eds.) *Simulation Modeling and Analysis with ARENA*. Burlington: Academic Press. ISBN 978-0-12-370523-5, pp. 11–21. DOI:https://doi.org/10.1016/B978-012370523-5/50003-1. URL https://www.sciencedirect.com/science/article/pii/B9780123705235500031.

Augusto A, Conforti R, Dumas M and La Rosa M (2017) Split miner: Discovering accurate and simple business process models from event logs. In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 1–10.

Berti A, van Zelst SJ and van der Aalst WMP (2019) Process mining for python (pm4py): Bridging the gap between process- and data science. *CoRR* abs/1905.06169. URL http://arxiv.org/abs/1905.06169.

Bradley W and Henseler J (2007) Modeling reflective higher-order constructs using three approaches with pls path modeling: a monte carlo comparison .

Camargo M, Dumas M and González-Rojas O (2019a) *Learning Accurate LSTM Models of Business Processes*. ISBN 978-3-030-26618-9, pp. 286–302. DOI: 10.1007/978-3-030-26619-6_19.

Camargo M, Dumas M and Rojas OG (2019b) Simod: A tool for automated discovery of business process simulation models. In: *BPM (PhD/Demos)*. pp. 139–143.

Ching W, Ng MK and Zhang S (2005) On computation with higher-order markov chains. In: Zhang W, Tong W, Chen Z and Glowinski R (eds.) *Current Trends in High Performance Computing and Its Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-27912-9, pp. 15–24.

Ching WK, Fung ES and Ng MK (2004) Higher-order markov chain models for categorical data sequences. *Naval Research Logistics (NRL)* 51(4): 557–574.

development team P (2020) pandas-dev/pandas: Pandas. DOI: 10.5281/zenodo.3509134. URL https://doi.org/10.5281/zenodo.3509134.

Dumas M, La Rosa M, Mendling J, Reijers HA et al. (2018) *Fundamentals of business process management*, volume 2. Springer.

Efron B, Hastie T, Johnstone I and Tibshirani R (2004) Least angle regression. *The Annals of Statistics* 32(2). DOI:10.1214/009053604000000067. URL http://dx.doi.org/10.1214/009053604000000067.

Evermann J, Rehse JR and Fettke P (2016) A deep learning approach for predicting process behaviour at runtime. *International Conference on Business Process Management* 1: 490. DOI:10.1007/978-3-319-58457-7. URL http://b-ok.xyz/book/2942192/1d94cd.

Fishman GS (2001) *Discrete-event simulation: modeling, programming, and analysis*. Berlin: Springer-Verlag. DOI:10.1017/978-1-4757-3552-9.

Fracca C, Bianconi A, Meneghello F, de Leoni M, Asnicar F and Turco A (2021) Bpsimpy: A python library for wfmc-standard process-simulation specifications. In: *BPM (PhD/Demos)*. pp. 97–101.

Grüger J, Geyer T, Jilg D and Bergmann R (2022) Sample: A semantic approach for multi-perspective event log generation-research paper. In: *5th International Workshop on Process-Oriented Data Science for Healthcare (PODS4H22)*.

Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C and Oliphant TE (2020) Array programming with NumPy. *Nature* 585(7825): 357–362. DOI:10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Hochreiter S and Schmidhuber J (1997) Long short-term memory. *Neural computation* 9(8): 1735–1780.

Iverson KE (1962) *A Programming Language*. USA: John Wiley Sons, Inc. ISBN 0471430145.

Kubrak K, Milani F, Nolte A and Dumas M (2022) Prescriptive process monitoring: Quo vadis? *PeerJ Computer Science* 8: e1097.

La Rosa M and Soffer P (2013) *Business Process Management Workshops: BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012, Revised Papers*, volume 132. Springer.

López-Pintado O and Dumas M (2022) Business process simulation with differentiated resources: Does it make a difference? In: *Business Process Management: 20th International Conference, BPM 2022, Münster, Germany, September 11–16, 2022, Proceedings*. Springer, pp. 361–378.

Mannhardt F and Blinde D (2017) Analyzing the trajectories of patients with sepsis using process mining. *RADAR+ EMISA@ CAiSE* 1859: 72–80.

Mannhardt F, de Leoni M, Reijers H and Aalst W (2015) Balanced multi-perspective checking of process conformance. *Computing* DOI:10.1007/s00607-015-0441-1.

Navarin N, Vincenzi B, Polato M and Sperduti A (2018) LSTM networks for data-aware remaining time prediction of business process instances. *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings* 2018-Janua: 1–7. DOI:10.1109/SSCI.2017.8285184.

Niek Tax, Marlon dumas, Ilya veenich, Marcello la rosa (2017) Predictive Business Process Monitoring with LSTM Neural Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10253 LNCS: V–VI. DOI:10.1007/978-3-319-59536-8.

Peeperkorn J, vanden Broucke S and De Weerdt J (2022) Can deep neural networks learn process model structure? an assessment framework and analysis. In: *Process Mining Workshops: ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31–November 4, 2021, Revised Selected Papers*. Springer, pp. 127–139.

Pegoraro M, Uysal MS and van der Aalst WM (2021) Proved: A tool for graph representation and analysis of uncertain event data. In: *Application and Theory of Petri Nets and Concurrency: 42nd International Conference, PETRI NETS 2021, Virtual Event, June 23–25, 2021, Proceedings 42*. Springer, pp. 476–486.

Pourbafrani M, Jiao S and van der Aalst WM (2021a) Simpt: process improvement using interactive simulation of time-aware process trees. In: *Research Challenges in Information Science: 15th International Conference, RCIS 2021, Limassol, Cyprus, May 11–14, 2021, Proceedings*. Springer, pp. 588–594.

Pourbafrani M and van der Aalst WM (2020) Pmsd: data-driven simulation using system dynamics and process mining. *arXiv preprint arXiv:2010.00943* .

Pourbafrani M, Vasudevan S, Zafar F, Xingran Y, Singh R and van der Aalst WM (2021b) A python extension to simulate petri nets in process mining. *arXiv preprint arXiv:2102.08774* .

Raftery AE (1985) A model for high-order markov chains. *Journal of the Royal Statistical Society. Series B (Methodological)* 47(3): 528–539. URL http://www.jstor.org/stable/2345788.

Railsback SF, Lytinen SL and Jackson SK (2006) Agent-based simulation platforms: Review and development recommendations. *Simulation* 82(9): 609–623.

Ratzer AV, Wells L, Lassen HM, Laursen M, Qvortrup JF, Stissing MS, Westergaard M, Christensen S and Jensen K (2003) Cpn tools for editing, simulating, and analysing coloured

petri nets. In: *Applications and Theory of Petri Nets 2003: 24th International Conference, ICATPN 2003 Eindhoven, The Netherlands, June 23–27, 2003 Proceedings*. Springer, pp. 450–462.

Rubinstein RY and Kroese DP (2016) *Simulation and the Monte Carlo Method*. 3rd edition. Wiley Publishing. ISBN 1118632168.

Schreiber J (2018) Pomegranate: fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research* 18(164): 1–6.

Teinemaa I, Dumas M, Leontjeva A and Maggi FM (2018) Temporal stability in predictive process monitoring. *Data Mining and Knowledge Discovery* 32(5): 1306–1338. DOI: 10.1007/s10618-018-0575-9.

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

Teniente E and Weidlich M (2018) *Business Process Management Workshops: BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*, volume 308. Springer.

Tibshirani R (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58(1): 267–288. URL http://www.jstor.org/stable/2346178.

Trivedi KS and Bobbio A (2017) *Reliability and Availability Engineering: Modeling, Analysis, and Applications*. Cambridge University Press. DOI:10.1017/9781316163047.

van der Aalst WMP (2016) *Process Mining: Data Science in Action*. 2 edition. Heidelberg: Springer. ISBN 978-3-662-49850-7. DOI:10.1007/978-3-662-49851-4.

van Dongen BF, Crooy RA and van der Aalst WM (2008) Cycle time prediction: When will this case finally be finished? In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, pp. 319–336.

Verenich I, Dumas M, Rosa ML, Maggi FM and Teinemaa I (2019) Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *ACM Transactions on Intelligent Systems and Technology* 10(4): 1–34. DOI:10.1145/3331449.

Zou H and Hastie T (2005) Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 67(2): 301–320. URL http://www.jstor.org/stable/3647580.

Paper III

Riess, M. and Scholderer, J. (2023). Customer-service queuing based on predicted loyalty outcomes. Manuscript submitted to *Decision Support Systems*.

CUSTOMER-SERVICE QUEUING BASED ON PREDICTED LOYALTY OUTCOMES

Mike Riess
School of Economics and Business
Norwegian University of Life Sciences
Universitetstunet 3, 1433 Ås, Norway
mike.riess@nmbu.no

Joachim Scholderer
School of Economics and Business
Norwegian University of Life Sciences
Universitetstunet 3; 1433 Ås, Norway
joachim.scholderer@nmbu.no

ABSTRACT

We introduce a customer service management approach with a new type of predictive priority queuing. The priority rank of a case is determined based on predicted customer loyalty scores. Increases in customer loyalty are predicted based on predicted case throughput time. Case throughput time is, in turn, predicted from seasonality indicators alone, due to incomplete information at the time of entering the queue. The priority queue is continuously updated. Two studies are reported. Study 1 is a statistical analysis of two years of customer service data from a European internet and telecommunications provider. We utilise the real-world data to obtain valid estimates of the parameters of all distributions governing the service process. In Study 2, we use the calibrated distribution parameters in Monte Carlo simulations of the customer service management system. In these experiments, we compare our predictive priority queuing approach with the traditional first come, first served approach used in the case company. Furthermore, we compare the performance to two other approaches that are based on predicted case throughout time; the longest remaining time first and the shortest remaining time first approach. We show that prioritisation based on predicted customer loyalty does indeed lead to an increase in average customer loyalty. However, since this effect is mediated by predicted throughput time, the longest remaining time first approach yields similar results as our proposed method. Introduction of a service level (here: maximum 60 hours waiting time) reduced the performance of all approaches based on predicted throughout time to the performance of the first come, first served approach.

1 Introduction

In a classical experiment, Kumar et al. (1997) show that service time guarantees can be a two-edged sword: customers will only be more satisfied at the end of a waiting period if the guaranteed service times are actually met. If the guarantees are not met, however, customers will be even less satisfied than if they had not been aware of the existence of a service time guarantee in the first place. This poses a challenge for customer service management (Ibrahim, 2018). Since service processes are interactive by nature, depending to a large degree on actions by the customer, they can never be fully under the control of the service provider. Compared to manufacturing processes, a much larger proportion of cases will be outside the "specification limits" defined by waiting time guarantees (Reijers, 2003). The problem is known and has been addressed in several ways in previous research.

Service management approaches based on queuing theory, for example Hui and Tse (1996), Obermeier et al. (2020) or Schwarz et al. (2016), reduce the between-case variance of the waiting time distribution and thereby also the number of cases exceeding a given waiting time guarantee. In its simplest form, a queuing system treats newly arriving cases as equal in terms of the gains associated with meeting a waiting time guarantee or the losses associated with exceeding a waiting time guarantee. This is for example the case with the standard first come, first served (FCFS) discipline. More elaborate approaches introduce classes of cases or customers in order to prioritise the queue (Sayenko et al., 2006). Class membership can be static and known before case arrival (e.g., based on customer tenure, customer lifetime value or a related metric). Alternatively, the class membership of a case may be dynamic but known at the time the queue is updated (e.g., based on current prices of spare parts or repair activities). Finally, it can be dynamic and unknown at the

time the queue is updated. In this situation, a prediction at case level is required of the quantity (or quantities) based on which priority class membership is to be assigned.

In the research presented here, we shall introduce such an approach. We study an $M/M/s$ system with priority queuing. The priority rank of a case is determined based on predicted increases in customer loyalty. Increases in customer loyalty are predicted based on predicted case throughput time. Case throughput time is, in turn, predicted from seasonality indicators alone, as no case attributes are available at the time of prioritisation. The priority queue is updated every time a new case arrives. Using a simulation approach calibrated on two years of real-world customer service data, we show that prioritisation based on predicted increases in customer loyalty does indeed lead to an increase in average customer loyalty (compared to first come, first served). In addition, it also leads to lower average throughput time. Both effects are relatively stronger when the number of customer service agents s is lower. We also compare our approach to two other queuing disciplines two other approaches that are based on predicted case throughout time, including the longest remaining time first approach (LRTF) and the shortest remaining time first approach (SRTF). Compared to these, our loyalty-based approach still yields higher average customer loyalty, but only is only slightly better than SRTF, and equal to LRTF, in terms of case throughput time. Introduction of a service level (here: maximum 60 hours waiting time) to increase the fairness of the systems reduced the performance of all approaches based on predicted throughout time to the performance of the first come, first served approach used in the case company.

To our knowledge, this is the first paper in the queuing literature that addresses *predicted loyalty outcomes* in its prioritisation approach. The paper is structured as follows. In the remainder of the introduction section, we will review relevant theory and previous research, describe the rationale of our approach, and explain our research question and the case background for the empirical part. Then, we will report Study 1, a statistical analysis of two years of real-world customer service data from a European internet and telecommunications services provider. We will document the estimation of all distribution parameters needed in the simulation and the development of the statistical prediction models. In Study 2, we will use the calibrated distribution parameters and statistical models in simulations of our $M/M/s$ system, comparing our prediction-based prioritisation approach with the traditional FCFS approach and two other approaches based on predicted case throughout time: The SRTF and the LRTF approach.

1.1 Theory and previous research

1.1.1 Prioritisation approaches in queuing

In the terminology of Kendall (1953), the "discipline" of a queuing system denotes the order in which the cases in the queue are processed. Traditional approaches include first come, first served (FCFS), last in, first out (LIFO) and shortest job first (SJF). *Priority queuing* is used when the arriving cases differ in terms of their value to the business. Many prioritisation approaches have been suggested in the literature. In principle, prioritisation can be based on any set of cost or value indicators that are available on the level of a case or customer. Gurvich et al. (2008), for example, study the impact of different prioritisation approaches in service systems. They propose a model based on an $M/M/s/N$ queuing system with the goal to find the optimal staffing s, given customer classes with different processing times, subject to a desired service level. The queue is prioritised using an idle-server based threshold priority and their proposed single-class staffing rule. The authors find that there is a decoupling between staffing and prioritisation approach when they evaluate a multi-class system, such that staffing in this setting has similar dynamics to a single class system.

In another paper, Dobson and Sainathan (2011) study the impact of prioritised queue management under the assumption that prioritisation comes with a cost. Their approach features two types of agents: sorters and processors. The role of the sorters is to gather information and prioritise each of the incoming customers, while the processors perform the work needed on the cases. This introduces an additional cost related to the sorter agents, but assuming heterogeneous waiting costs, sorting is expected to have a net positive effect on total costs. The results show that, given a fixed exogenous budget, this prioritisation will only outperform FCFS in some cases, depending on the ratio of sorting cost to processing cost. Mahmoumgonbadi et al. (2019) propose a queue prioritisation method based on a fuzzy system. In theory, the approach would assign weights to customers that are based on service duration, service value, customer tenure, customer patience, and waiting time. Based on (not empirically calibrated) simulation experiments, the authors find that their approach outperforms FCFS in an $M/M/1$ system with different levels of stability.

The work of Tan et al. (2012) utilises the shortest remaining time first (SRTF) discipline in a case study of a hospital emergency department. To improve the patient flow, three variants of the SRTF discipline are compared to FCFS in calibrated simulation experiments of a $M/M/s$ queue. The authors find that the SRTF-based strategies lead to shorter throughput times. However, this depended heavily on the accuracy of the remaining time prediction. Furthermore, the authors report that generic SRTF had the disadvantage of leading to starvation, i.e. the unfair situation that some customers/patients will wait indefinitely in the queue). In Wang et al. (2020) the authors study the application of multiple variants of the longest remaining time first discipline (LRTF) in the context of edge computing server allocation.

Four different variants of the LRTF discipline are compared to the FCFS in a $M/M/s$ queue. The LRTF discipline led to the shortest average response times, compared to FCFS. However,in a similar manner as the SRTF discipline, generic LRTF suffers from the problem of starvation.

In the research presented in this paper, we will compare our proposed approach (NPS) to three competitors: the queue discipline most commonly applied in customer service systems (FCFS) plus two disciplines which are, like our proposed approach, based on predicted throughput times (SRTF and LRTF) but do not use the throughout time predictions in further prediction steps. We will compare the four disciplines in scenarios with and without waiting time guarantees in the form of a *service level*. Service levels are often introduced in customer service systems to avoid the starvation problems which disciplines such as SRTF and LRTF are known for (see above) and thereby to increase the fairness of priority-based queuing systems.

We will not consider preemptive queuing approaches in our comparisons. The concept refers to a group of queue disciplines (e.g., see Segal (1970) where a customer who arrives while the server is busy serving another customer is given priority over the existing customer. The server interrupts the ongoing service to attend to the new customer, and then resumes the previous service where they left off. This approach has some theoretical advantages, but it is rarely used in customer service queuing systems. There are two reasons for this. First, customer service queuing systems are designed to provide fair and equitable service to all customers. Preemptive queuing strongly favors high-priority customers at the expense of low-priority customers, and often in quite obvious ways, which can lead to dissatisfaction and complaints. Second, customer service queuing systems are usually designed to optimise resource utilisation and minimise costs. Due to the frequent task-switching, preemptive queuing can increase the complexity and cost of the system, which may not be justifiable for the marginal gains in efficiency or responsiveness.

1.1.2 Customer loyalty as a prioritisation criterion

To our knowledge, customer loyalty has so far not been used in any prioritisation approaches studied in the queuing literature. This is somewhat surprising, considering that customer loyalty is the central construct in all relationship-oriented approaches to marketing and service management (Hallowell, 1996; Kumar and Shah, 2004; Sheth and Parvatiyar, 1995): the more loyal a customer, the higher that customer's lifetime value, and the more loyal the customers of a company, the higher the company's expected future profitability. The commonly accepted definition of customer loyalty is "the biased (non-random) behavioural response (brand support), expressed over time by some decision-making unit with respect to one or more alternative brands out of a set of such brands, and is a function of psychological processes (brand commitment)" (Jacoby and Chestnut, 1978).

Whilst there is little dispute about the conceptual definition and the importance of loyalty, its operationalisation differs vastly in practice. Systematic reviews distinguish two basic approaches to the measurement of customer loyalty (Bennett and Rundle-Thiele, 2002; Jacoby and Chestnut, 1978; Knox and Walker, 2001; Rundle-Thiele, 2005; Watson et al., 2015): behavioural approaches and attitudinal approaches. Behavioural approaches to the measurement of loyalty use objective indicators. Typical measures are the share of purchases of a brand relative to the total purchases within the respective category, measures of allegiance (i.e., how long the customer has stayed with the supplier), and the number of competing suppliers a customer has relations with. Attitudinal approaches to the measurement of loyalty, on the other hand, use measures of commitment and intention, for example intentions to remain a customer, re-purchase the product or service, or recommend the product or service to others.

Unfortunately, attitudinal and behavioural measures of loyalty have often shown limited convergent validity. In a cross-category study of brand loyalty in consumer goods markets, Chaudhuri and Holbrook (2001) found a correlation of 0.64 between the two alternative measures. In a similar study, Knox and Walker (2003) found a much lower correlation of 0.23. In a study of loyalty in telecommunications service markets, Rundle-Thiele and Mackay (2001) found correlations ranging from 0.10 to 0.50. Since its first publication by Bain consultant Frederick Reichheld (Reichheld, 2003), the *net promoter score* (NPS) has come to replace virtually all other attitudinal measures of customer loyalty. The NPS is based on a single survey question that measures word-of-mouth intention ("How likely is it that you would recommend [company or brand] to a friend or colleague?") on an 11-point Likert scale ranging from "not at all likely" (0) to "extremely likely" (10). At the analysis stage, the raw scale score is transformed into a three-class variable: customers who responded with levels 9 or 10 are labelled *promoters*, customers who responded with levels 6 or lower are labelled *detractors*, and customers who responded with levels 7 or 8 are labelled *passives*. The transformation is non-linear and follows the logic of the top-box rule in market research (Kalwani and Silk, 1982), acknowledging that only differences in the upper regions of intention scales have predictive validity with respect to differences in the probability of the target behaviour.

Due to their as-yet undecided status, the passives are often considered the segment where targeted customer relationship management activities can have the highest leverage and impact. Converting passives to promoters would require no more than a shift of one or two scale points upwards on the response scale of the NPS survey question. In a similar way,

passives could become detractors by a shift of one or two additional scale points downwards on the response scale. Hence, the passives can be regarded as the most sensitive segment of customers. We adopt this way of thinking in the prioritisation approach investigated here.

Let i $(i = 1, 2, ...N)$ be an active case in queue Θ at time t $(t = 1, 2, ...T)$. Furthermore, let $\hat{NPS}_i$ be the predicted NPS at time $t = T$ of the customer who submitted i, that is, after the service has been completed and the case has been resolved. The priority rank of case i is then determined by rank-transforming the distance of the predicted after-service NPS of the customer to the midpoint of the interval on the net promoter score response scale that represents the segment of the passives. This midpoint is $NPS = 7.5$, such that:

$$priority(i, t) = rank(|\hat{NPS}_{i,T} - 7.5|) \tag{1}$$

The rank function maps the set of NT distinct values to the set of integers $1, 2, ..., N$ such that each value is assigned a unique rank based on its position in the ordered list of values (i.e., the smallest value receives a rank of 1, the second smallest receives a rank of 2, and so on). The variable *priority* is therefore ordinal: a value of 1 indicates that case i has first priority at time t, a value of 2 indicates that case i has second priority at time t, and so on.

Since customer satisfaction and loyalty are partly determined by the customer's experience with the company's service provision, they will partly depend on the time it will take the company to resolve the case (Bielen and Demoulin, 2007; Djelassi et al., 2018; Ibrahim and Whitt, 2011; Kumar et al., 1997; Tom and Lucey, 1997). Since the throughput time of the case is not known before the case has been resolved, a prediction of the throughout time for case i must be made before the after-service customer loyalty $\hat{NPS}_{i,T}$ can be predicted.

1.2 Research questions

The objective of the research presented here is to develop a new approach to customer service management that addresses *customer loyalty* in its queue prioritisation approach, as well as evaluate its suitability via a case study. To achieve this, we divide the research into two studies:

In Study 1, we will document the estimation of all distribution parameters needed in a simulation model of the case company (which will be introduced in the next section). Furthermore, we will document the estimation of the needed prediction models in order to perform the proposed loyalty-based queue prioritisation. The work is guided by the following research question:

- **RQ1.1:** What are the distribution parameters of the key process components needed to generate an agent-based simulation model of the service process in the case company?

In Study 2, we will use the calibrated distribution parameters and statistical models in Monte Carlo simulations, comparing the NPS-based predictive priority queuing approach with three other queue disciplines (FCFS, LRTF, SRTF), guided by the following research questions:

- **RQ2.1:** How does the number of agents influence the queue waiting time in the simulation period under different prioritisation schemes?
- **RQ2.2:** What are the effects of the four queue disciplines on overall process performance?
- **RQ2.3:** What are the effects of the proposed loyalty-based queue discipline on the simulated net promoter score?

A conceptual overview of the two studies is shown in Figure 1. As illustrated in the figure, Study 1 utilises event log data provided by the case company which will be further introduced in section 1.3. Since neither case throughput time nor after-service customer loyalty are known at the point in time where cases must be prioritised, our approach involves two statistical prediction models: (a) a model that predicts expected throughput time, given case attributes that are known at the time the case enters the queue, and (b) a model that predicts customer loyalty, given throughput time and case attributes that are known at the time the case enters the queue. In each discrete time unit, both models are used to update the ordering of cases in the queue.

We see it as important that the merit of our approach is assessed based on as realistic assumptions as possible. Hence, the values of all simulation parameters will be based on a real business case. The case context as such is described in the following section.

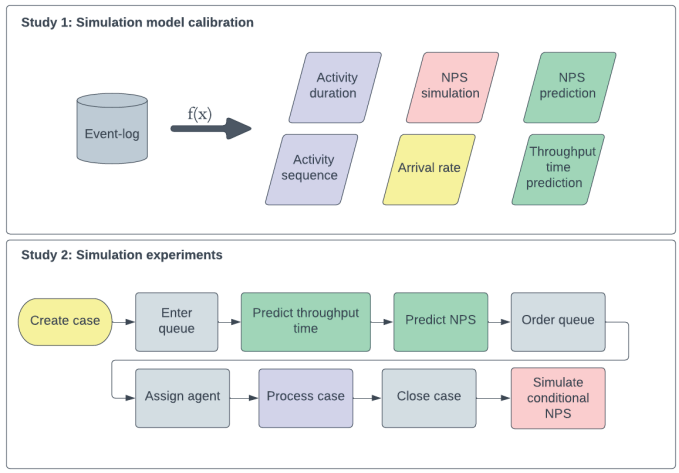


Figure 1: *Conceptual overview of the two studies.*

1.3 Case context

The case company is a European internet and telecommunications services provider that wishes to remain anonymous. The case company uses a customer relationship management (CRM) system for the management of customer service requests from its enterprise customers. In the study period of almost two years (February 2018 to December 2019), the case company had 91 different agents working on 11,294 email-based service requests in total.

The agents are based at different physical locations, but have remote access to the same systems to help solve customer inquiries. Agents are organised into multiple teams, and do not exclusively work on customer service inquiries at all times. Customers can contact the company via e-mail, telephone or a customer support request form on the company's website. Contact via telephone is mainly utilised by medium and large enterprise VIP customers (these have their own dedicated key account manager and thereby represent a minority of the customer base). Once a case is created, it is put in the queue until it is assigned to an available agent. Due to the form of contact, the case topic is either missing or unverified at the time the agent is assigned. Once an agent has been assigned, he or she will work on the case and have correspondence with the customer via e-mail or telephone, in case further information is needed to resolve the issue. When the issue is resolved, the case is closed.

The case company uses the Net promoter score (NPS) Reichheld (2003) as a tool to evaluate its performance after a case has been closed. The customer receives a short text message (SMS) and is asked to reply on a scale from 0 to 10, how likely he or she is to recommend the case company to others. Of all 11294 cases in the study period, 1897 received a valid NPS reply (16.7% response rate). The average individual NPS response was 8.82. The issue-to-resolution process at the case company is visualised as a BPMN model in Figure 2.

2 Study 1: Statistical analysis of real-world data

Study 1 is a statistical analysis of the customer service process that will be simulated in Study 2. The purpose of the analysis is to obtain valid estimates of all distribution parameters governing the process: activity duration, activity sequences, inter-arrival times, case throughput time, and customer loyalty (operationalised in terms of the net promoter score). The main motivation for modelling activity duration, activity sequences and inter-arrival times is to enable good calibration to the real-world customer service process in the case company. The throughput time model is a key component in the predictive prioritisation approach used for queue management in Study 2. The net promoter score, which is dependent on throughput time, will be modelled in two variants: (a) including all relevant information, including information that will only be known once a case is completed, (b) using the predicted throughput time only. This will be further explained in Study 2.

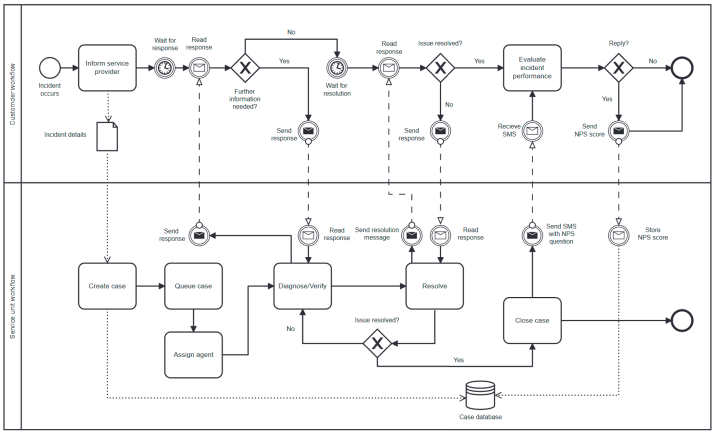


Figure 2: *BPMN model of the customer service process in the case company.*

2.1 Method

2.1.1 Data

The analysis will be performed on a subset of the data (1897 out of 11294 cases) from the module of the case company's Salesforce™ CRM system that is used for customer service management. All cases are created via email, and the subset contains only cases where, after the completion of the case, the relevant customer responded to the SMS request to answer the net promoter score (NPS) question. The data is in event log format (see Van der Aalst, 2016) and consists of 5456 events related to 1897 cases. The event log was created by combining the Salesforce™ *case* and *task* objects. The case object contained static features such as the responsible agent, as well as the case topic and time stamps. The task object included the events related to the cases. At this level, an *interaction* with a customer denotes a phone call, whereas e-mail correspondence is explicitly coded as *email*.

2.2 Results

In the following, the key characteristics of the customer service process in the case company will be modelled using statistical techniques.

2.2.1 Activity duration

Modelling the conditional activity duration is a crucial input to the Monte Carlo simulations in Study 2 where the simulated throughput time will be modelled as a function of the waiting time plus the individual activity durations in a given case.

Activity duration was modelled using a set of model candidates from the generalised linear model family with different link functions and regularization penalties. An overview of the fitted models is shown in Table 8. The selected model was a generalised linear model with a Weibull link function, which yielded a generalised $R^2 = 0.21$ on the training set (70-30 split) and a generalised $R^2 = 0.19$ on the test set. The prediction expression has the following form:

$$\hat{duration} = \exp\left(\alpha + \sum_{i=1}^{n} X_i\beta_i\right)\Gamma(1+\theta) \tag{2}$$

where X_i denotes the ith input feature, β_i the ith model parameter, Γ the *gamma-function*, and θ the shape parameter of the gamma distribution. The simulation equation is of the form:

Table 1: Model coefficients: Weibull regression (target variable: activity duration in hours; square brackets represent levels of categorical factors; the last level is the reference level)

Term	$\hat{\beta}$	SE	p
Intercept	1.6645	0.1650	<.0001
case_topic[d_2-z_4]	0.0200	0.0456	0.6620
case_topic[g_1-z_4]	-0.0538	0.0313	0.0857
case_topic[j_1-z_4]	-0.0557	0.0213	0.0088
case_topic[q_3-z_4]	0.1712	0.0587	0.0035
case_topic[r_2-z_4]	0.0836	0.0383	0.0288
case_topic[w_1-z_4]	-0.0609	0.0286	0.0334
case_topic[w_2-z_4]	0.0119	0.0313	0.7043
case_topic[z_2-z_4]	-0.0420	0.0433	0.3319
case_topic[z_3-z_4]	0.1637	0.0369	<.0001
task_tasksubtype[Email-Task-Reminder]	0.0180	0.1158	0.8767
task_tasksubtype[Interaction-Task-Reminder]	0.1057	0.1319	0.4229
task_number	0.0420	0.0177	0.0176
resource	0.2171		
theta	0.3908		

Table 2: Transition matrix: Absorbing Markov chain.

	Task-Reminder	Interaction	Email	END
Task-Reminder	0.08	0.0	0.67	0.25
Interaction	0.0	0.02	0.96	0.02
Email	0.0	0.02	0.45	0.53
END	0.0	0.0	0.0	1.0

$$duration = Weibull(\frac{1}{\theta}, \exp(\alpha + \sum_{i=1}^{n} X_i \beta_i)) \tag{3}$$

The model included features representing four input variables: *case topic*, *activity*, *activity number* and *resource*. Case topic refers to one of ten anonymised categories of customer service cases distinguished by the case company. Activity refers to one of three types of activities logged in the CRM system (see Section 2.1.1). Activity number is the order of the given activity in the trace. Resource is a categorical variable with 77 different levels referring to the individual agents observed in the two-year span of the data. This particular variable was forced into the model without regularisation in order to ensure that the effect of the individual agent (reflecting different speeds) would be represented in the model.

To enable further generalisation in the manner of a random effect, a normal distribution was fitted to the individual model coefficients representing the 77 different resource levels, with mean $\mu = 0.22$ and standard deviation $\sigma = 0.52$. The individual effects of the agents in Study 2 will therefore be simulated from the fitted distribution, rather than the individual resources observed in the data.

The coefficients of the model are presented in Table 1. Since the *resource* effect is modelled using a normal distribution, its 77 original parameter values are excluded from the coefficients in Table 1. However, the full table with all parameters can be found in Appendix 9.

2.2.2 Activity sequences

The *trace* or sequence of activities in a given case is modelled using an absorbing Markov chain. The model thus consists of $k + 1$ states, where the states are the possible activities plus the absorbing state, 'END'. Here, the absorbing state is added as the last step in each of the observed traces. The Markov chain prediction has the general form:

$$P(X_{t+1} \mid X_1, X_2, ..., X_t) = P(X_{t+1} \mid X_t) \tag{4}$$

The transition probabilities P were estimated using maximum likelihood and are shown in Table 2, whilst the initial probabilities P^0 are shown in Table 3.

Table 3: Initial probabilities

	Task-Reminder	Interaction	Email	END
P^0	0.0	0.92	0.08	0.0

Table 4: Model coefficients: exponential regression (target variable: hours to next case arrival)

Term	$\hat{\beta}$	SE	p
Intercept	726.6267	323.9276	0.0249
year	-0.3589	0.1605	0.0253
month	-0.0881	0.0232	0.0001
day	0.0078	0.0081	0.3308
weekday	0.2616	0.0473	<.0001

2.2.3 Inter-arrival times

The inter-arrival times are the times between the arrivals of two consecutive cases. Modelling this distribution is important for Study 2 where the inter-arrival times control the traffic intensity in the modelled system (here a customer service function), given different levels of staffing. As the simulation will run over a one-year period between 2018 and 2020, we include seasonality patterns in the model.

A set of generalised linear models with different link functions was estimated. The candidate models are shown in Table 12 in the Appendix. The selected model was a generalised linear model with an exponential link function, which yielded a generalised $R^2 = 0.33$ in the training set (70% of the data) and a generalised $R^2 = 0.34$ in the validation set (30% of the data). The prediction expression has the following form:

$$\hat{InterArrivalTime} = \exp\left(\alpha + \sum_{i=1}^{n} X_i \beta_i\right) \tag{5}$$

The simulation equation is:

$$InterArrivalTime = -\log(1 - U(0,1)) \times \exp\left(\alpha + \sum_{i=1}^{n} X_i \beta_i\right) \tag{6}$$

Where $U(0,1)$ represents a scalar from the uniform distribution with $min = 0$ and $max = 1$. The model coefficients are shown in Table 4

2.2.4 Throughput time

The main purpose of modelling case throughput time is to provide additional information about a given case when it has not yet been assigned. This information will be used to model the NPS, which is used to prioritise the cases. A set of candidate models was estimated (see Table 13). The selected model was a generalised linear model with an exponential link function, which yielded a generalised $R^2 = 0.055$ in the training set (70% of the data) and a generalised $R^2 = 0.053$ in the validation set (the remaining 30% of the data). The target was modelled with an offset of 1. The prediction expression has the form:

$$\hat{Throughput\_time} = exp(\alpha + \sum_{i=1}^{n} X_i \beta_i) - 1 \tag{7}$$

Estimates of the model coefficients are shown in Table 5. The features represent the date and time of a case's arrival in the queue. We deliberately used simple linear approximations to the nonlinear effects of hour of day, weekday, and day of month in order to make the model more generalisable beyond this particular customer service group in our case company. The linear features model more general effects such as different arrival rates and case resolution times at the beginning versus end of work day and work week (Pope, 2016; Wieth and Zacks, 2011; Yao et al., 2019). Even though this approximate model may have low precision in its predictions of specific durations, these predictions will still be sufficient for assigning priority ranks to cases. In the decision making literature, this property is known as the "robust beauty of improper linear models" (Dawes, 1979).

Table 5: Model coefficients: exponential regression (target variable: case throughput time in minutes)

Term	$\hat{\beta}$	SE	p
Intercept	66.809	355.374	0.851
year	-0.030	0.176	0.867
month	-0.041	0.028	0.144
day	0.000	0.000	1.000
weekday	0.000	0.056	0.995
hour	0.065	0.052	0.211

Table 6: Model coefficients: gamma regression (target variable: NPS)

Term	$\hat{\beta}$	SE	p
Intercept	2.3006	0.0430	<.0001
Log[case_throughputtime+1]	-0.0098	0.0052	0.0590
case_topic[d_2-z_4]	-0.1291	0.1049	0.2183
case_topic[g_1-z_4]	0.1008	0.0481	0.0362
case_topic[j_1-z_4]	0.0853	0.0433	0.0490
case_topic[q_3-z_4]	-0.0427	0.0549	0.4359
case_topic[r_2-z_4]	-0.0317	0.0848	0.7088
case_topic[w_1-z_4]	0.0000	0.0000	1
case_topic[w_2-z_4]	0.0000	0.0000	1
case_topic[z_2-z_4]	0.0967	0.0509	0.0577
case_topic[z_3-z_4]	-0.0048	0.0484	0.9217

2.2.5 NPS

The NPS is modeled for two specific purposes: (a) Simulating the NPS given complete information about the case after it has been closed, and (b) generating an initial prediction of the expected NPS using incomplete information. This section will first present the conditional distribution which will be used for simulating the NPS given complete information. A set of candidate models were estimated from the family of generalised linear models with normal, gamma and exponential link functions. The results are shown in Table 14 in the Appendix. The selected generalised linear model with gamma link yielded a generalised $R^2 = 0.040$ in the training set (70% of the data) and $R^2 = 0.039$ in the validation set (the remaining 30% of the data). The target variable was modelled using an offset of 1 (as the gamma distribution cannot represent zero). The simulation equation has the form:

$$NPS = \Gamma(\frac{Exp(\alpha + \sum_{i=1}^{n} X_i \beta_i)}{\rho}, \rho) - 1 \tag{8}$$

Where $\Gamma(k, \rho)$ denote the gamma distribution with shape parameter k and scale parameter $\rho = 1.3005$. The shape parameter is conditional on the intercept α and the linear combination of the input features X and their associated coefficients β, which are shown in Table 6.

For the initial NPS prediction model, which will be used to prioritise incoming cases under incomplete information, the case topic will not be available before the case has left the queue. We therefore only use predicted throughput time to predict the NPS value. In this case, the best model candidate was linear as shown in Equation (9).

$$\hat{NPS} = (\alpha + x\beta) - 1 \tag{9}$$

This model yielded R^2 values of 0.01 and 0.03 in the training and validations sets, respectively. The model coefficients are shown in Table 7.

Table 7: Model coefficients: gamma regression (target variable: NPS)

Term	$\hat{\beta}$	SE	p
Intercept	10.2211	0.1111	<.0001
Log[case_throughputtime+1]	-0.0949	0.0249	0.0001

3 Study 2: Monte Carlo experiments

In Study 2, we will recreate the customer service process in the case company in a simulation model, using the results of Study 1 to calibrate the relevant model parameters. In Monte Carlo experiments, we then compare the performance of the loyalty-based predictive priority queuing approach suggested in this paper with the traditional first come, first served approach. An overview of the approach is shown in Figure 3.

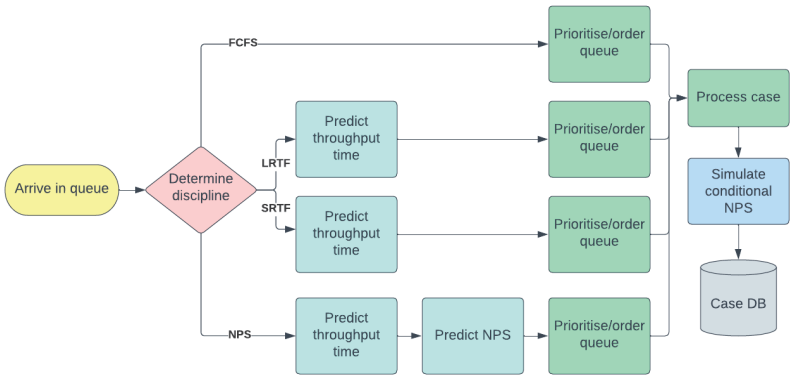


Figure 3: Overview of the competing queue management approaches.

For the loyalty-based queue management approach, a *priority score* is calculated every 15 minutes (update interval of queue management and case assignment) as: $Priority\_score = |7.5 - NPS_{arrival}|$. Here, $NPS_{arrival}$ refers to the predicted NPS-score (see 2.2.5) based on estimated throughput time (see Section 2.2.4) before the case has been assigned.

To evaluate the outcome of the experiments we also define NPS_{resp} which is the simulated conditional NPS-response, based on the case type and waiting time. To compare the two approaches, we run a set of simulation experiments spanning over 365 days in the period from 2018/07/01 to 2019/07/01.

3.1 Method

The method section is structured as follows: Firstly, the overall experimental design is presented in Section 3.1.1. Next, an overview of the simulation framework is presented. Sections 3.1.3 to 3.1.9 present details of the algorithms used in the experiments.

3.1.1 Experimental design

The simulation experiments were generated according to a full factorial $4 \times 7 \times 2$ design. The first factor was the queue discipline, the second the number of agents available, and the third the service policy. An overview of the three factors and their levels is shown below.

- **Factor 1: Queue discipline**
  - FCFS - First come, first served
  - SRTF - Shortest remaining time first
  - LRTF - Longest remaining time first
  - NPS - Net promoter score-based priority
- **Factor 2: Number of agents**
  - 3, 4, 5, 6, 7, 8, 9
- **Factor 3: Service level constraint**

- None
- Service level = 60 hours

For the queue discipline we use non-preemptive queuing and benchmark our NPS-approach against $FCFS$, $SRTF$ and $LRTF$. $FCFS$ is the current approach used in the case company and thus serve as the baseline. As no other attributes for case prioritisation are available when a case arrives (the topic category is only realised when an agent has read the email), the $SRTF$ and $LRTF$ approaches are implemented based on predicted remaining throughput time (Eq. 7). The $SRTF$ and $LRTF$ disciplines are included to provide contrast to our approach, illustrating two alternative predictive scheduling approaches using a similar input. The range in the number of agents is chosen to represent the contrast between sufficient and insufficient capacity in the case company. The capacity within a single simulation run is thereby assumed to be constant, as we wish to understand the interaction between the queue discipline, the capacity and the process performance. The service level constraint is included as a separate factor in order to reduce the issue of starvation in the NPS, $SRTF$ and $LRTF$ disciplines. In this case, it is assumed that a case must have left the queue within 72 hours to be compliant. The hard ceiling therefore takes effect after 60 hours, such that agents will have 12 hours to retrieve the case from the queue. When the hard ceiling takes effect, the previous order of the cases is overruled by the $FCFS$ discipline. Further details are shown in Section 3.1.5. A total of 100 replications were performed, resulting in a total of $N = 5600$ simulation runs. The source code used to run the experiments is publicly available[1].

3.1.2 Simulation procedure

The simulation was set up in such a way as to generate event stream data from a stochastic process that is similar in its nature and properties to the one that was empirically investigated in the case company in Study 1. All algorithms were calibrated using the statistical models and distributions fitted in Study 1 (see Section 2.1.1).

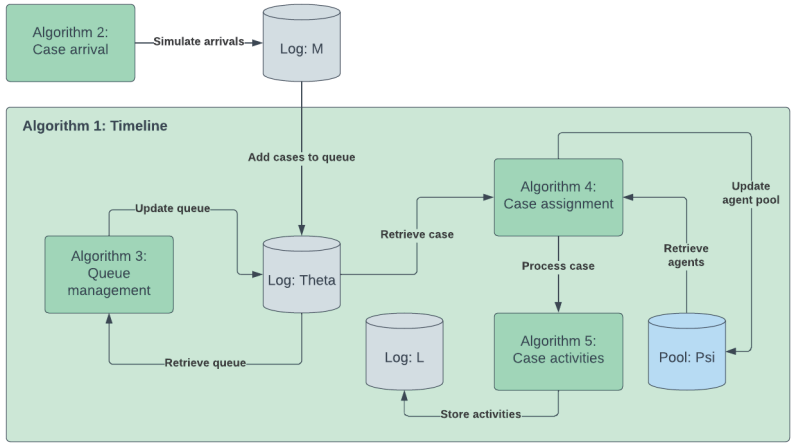


Figure 4: Overview of simulation procedure.

The general procedure of the simulation model is shown in Figure 4. The simulation model produces three event logs: M, Θ and L. Furthermore, an agent pool Ψ is defined. Firstly, the event log M is populated with all cases arriving within the simulation period using Algorithm 2. Next, the timeline is simulated using Algorithm 1. As the simulated timeline passes the dates of arrival of new cases, these are added to the event log Θ containing the cases in the queue. The queue is then updated by Algorithm 3, re-prioritising the order based on the given queue discipline. Cases in the queue are then assigned to an agent using Algorithm 4, which continuously updates the agent pool Ψ. Once a case is assigned to an agent, the case is processed using Algorithm 5 until the final activity is reached. The observed process behavior is then recorded in the final event log L. This process repeats itself until the end of the specified simulation period.

[1] github.com/Mikeriess/P3_queue_prioritization

3.1.3 Timeline

The simulation progresses in discrete time slices (days and 15-minute intervals). Within each day and case, activities can occur if and only if three criteria are met: (a) at least one open case must have activities pending, (b) at least one agents is available for performing activities, and (c) the current day is a workday. The timeline consist of a main loop that iterates over each day within a given range D_{end}. The arrival of new cases is controlled by the inter-arrival model fitted in Section 2.2.3. When cases are closed, they will get a simulated NPS-score based on their individual progress (using Eq. 8).

Algorithm 1: The discrete timeline simulation.

Data: queue discipline $P_{discipline} \subset \{FCFS, SRTF, LRTF, NPS\}$, number of days to simulate $D_{end} \in \mathbb{N}$, probability distributions $\omega \in \Omega$, feature mapping functions for conditional prediction models $\pi \in \Pi$, pool of case-handlers (agent objects) Ψ
Result: event log L, case-buffer Θ

```
/* Event log placeholders                                                    */
Θ, M, L ← ∅
/* Simulate the cases to arrive during the day                               */
M ← CaseArrival(i, D_end, π_arr, π_attr)
/* For each day D = (1, 2, .., D_end)                                        */
for d in D do
    /* Perform queue updates on a 15-minute window basis                     */
    z = d
    for every 15-minute window in day d do
        /* Add new cases to the case pool Θ and filter out cases that arrived after the
           15-minute window (and are therefore unobserved)                   */
        Θ ← Θ ⋃ M
        Θ ← {η ∈ Θ, |η_(q) ≤ z + 15}
        /* Sort case order, based on the queue discipline                    */
        Θ_ordered ← QueueManagement(Θ, P_discipline)
        /* Assign idle agents to newly arrived cases                         */
        Θ, Ψ ← CaseAssignment(Θ_ordered, Ψ)
        /* Perform activities on active cases                                */
        L, Θ, Ψ ← CaseActivities(d, Θ, Ψ, z)
        /* Update time by 15 minutes (0.010416 days)                         */
        z = z + 0.010416
        /* Simulate conditional NPS score for closed cases from Equation 8   */
        for η_(s=closed) in Θ do
            η_(c=NPS_resp) ← Model(η)
        end
    end
end
```

Algorithm 1 is the main simulation loop that iterates over the specified simulation periods $d = (1, 2, \ldots, D)$. Firstly, the set of all cases to arrive M in the simulation period is generated using Algorithm 2. Afterwards, the status of the cases and agents is updated every 15 minutes in the following order: First, the new cases that have arrived until the present point in time z, are added to the case buffer Θ. Next, queue management as described in Algorithm 3 is performed on Θ, and cases are prioritised based on the queue discipline. Next, idle agents in the agent pool Ψ are assigned to cases that are waiting in the queue (if any) by Algorithm 4. Finally, open cases with an agent assigned are processed by Algorithm 5. The resulting activities are written to the output event log L.

3.1.4 Case arrival

A case is represented as a trace in this simulation framework. A single trace η_i is defined as a multiset $\eta = (i, q, s, t, a, j, r, c)$, consisting of attributes of varying length: the case identifier i, the arrival-time q, the status of the case s, the set of timestamps t for the performed activities, the set of activities performed in chronological order a, the set of time indexes j for the performed activities, the set of agents r working on each of the activities, and finally the static case attributes (such as *case topic*, predicted *throughput time* and *NPS*) in the set c. The arrival time of a new

case is governed by the model in Equation 6, with parameter values as shown in Table 4. Implementation details are shown in pseudo code in Algorithm 1.

Algorithm 2: Case arrival.

Data: case identifier starting value i, number of days to simulate D_{end}, feature mapping function for inter-arrival time model π_{arr}, probability distribution for case attributes ω_{attr}
Result: set of newly arrived cases M, updated case identifier value i

```
/* Initialize the time variable z, and set of cases M                    */
```
$z = 0$
$M \leftarrow \emptyset$
```
/* Simulate today's case arrivals                                        */
```
while *current time* $z < D_{end} + 1$ **do**
```
    /* Simulate inter-arrival time q, from model in Equation 6           */
```
    $X \leftarrow \pi_{arr}(z, d)$
    $q = -\log(1 - U(0,1)) \times \exp\left(\alpha + \sum_{i=1}^{n} X_i \beta_i\right)$
```
    /* Generate a new case with arrival time q and assign a case identifier i  */
```
    $\eta \leftarrow (i, q, s \leftarrow \emptyset, t \leftarrow \emptyset, a \leftarrow \emptyset, j \leftarrow \emptyset, r \leftarrow \emptyset, c \leftarrow \emptyset)$
```
    /* Predict throughput time of the case from Equation 7               */
```
    $\eta_{(c=throughput)} \leftarrow Model(\eta_{(c=\{1,\ldots,k\})})$
```
    /* Predict NPS score of the case from Equation 9                     */
```
    $\eta_{(c=\hat{NPS}_{arrival})} \leftarrow Model(\eta_{(c=throughput)})$
```
    /* Append the case η to the set of cases M                           */
```
    $M \leftarrow M \bigcup \eta_i$
```
    /* Update time and case identifier variables                         */
```
    $z = z + q$
    $i = i + 1$
end

3.1.5 Queue management

Depending on the prioritisation approach, the queue will be treated differently. The $FCFS$ approach will leave the current order in the queue (Θ) unchanged, whereas $SRTF$ and $LRTF$ will order the queue by their predicted throughput time at arrival in ascending and descending order, respectively. The NPS approach will re-order the cases in the queue based on their predicted NPS score, as described in Section 1.1.2. *Passives* (customers answering 7-8 in the NPS survey question) will in this case be prioritised, as passives are *as yet undecided* and therefore more likely to change, i.e. to be either positively surprised (and thereby converted to promoters) or disappointed (and thereby become detractors), given a particular amount of resource expenditure. To ensure a certain degree of fairness, we also evaluate the effect of a hard ceiling, ensuring that customers put in the back of the queue will be prioritised once they have been there for more than 60 hours. This works as a two-step approach where the sorting based on the original queue discipline is firstly performed, after this step, all customers that have waited more than 60 hours will be moved in front of the queue (increasing the likelihood of a response within an assumed service level of 72 hours). The details of the prioritisation algorithms are shown in pseudo code in Algorithm 3.

3.1.6 Case assignment

Case assignment is performed for the agent pool $\Psi = (\psi_1, \ldots, \psi_k)$, which consist of all agents in the simulated customer service unit. First, an idle-pool A is created, such that only idle agents are assigned to new cases. Then, each agent in A is assigned to cases in $\Theta_{ordered}$ that are not currently assigned to an agent.

3.1.7 Case activities

Case activities are updated every 15 minutes as illustrated in Algorithm 1. In each update, each agent in the agent pool Ψ processes new activities if and only if two conditions are met: (a) the agent is assigned to a case and (b) the finish time for the current activity is within the current 15-minute window. When these conditions are met, a start delay v (Algorithm 6) is calculated if work on the next activity would be scheduled to begin outside business hours. In this case, the time until the next possible activity start *during* business hours is accumulated in v and subsequently added to the simulated activity duration. Next, Algorithm 7 simulates the next activity $a_{(j+1)}$ with timestamp $t_{(j+1)}$. The new activity and its timestamp are appended to the event log L. If the next activity is the absorbing state (END) of the

Algorithm 3: Queue management.

Data: case buffer Θ, queue discipline $P_{discipline} \subset \{FCFS, SRTF, LRTF, NPS\}$, and service level in days $P_{SLA} \in \mathbb{R}$
Result: updated case buffer $\Theta_{ordered}$
/* Only cases that arrived up until time $z + 0.010416$ are visible */
if $P_{discipline} = FCFS$ **then**
  /* Sort cases by their arrival time q */
  $\Theta_{ordered} \leftarrow Sort(\Theta, \Theta_{(c=q)}, ascending)$
if $P_{discipline} = SRTF$ **then**
  /* Sort cases in ascending order by their predicted throughput time $\hat{y}$ */
  $\Theta_{ordered} \leftarrow Sort(\Theta, \Theta_{(c=\hat{y})}, ascending)$
if $P_{discipline} = LRTF$ **then**
  /* Sort cases in descending order by their predicted throughput time $\hat{y}$ */
  $\Theta_{ordered} \leftarrow Sort(\Theta, \Theta_{(c=\hat{y})}, descending)$
if $P_{discipline} = NPS$ **then**
  /* Proceed if there is more than one case and a difference in predicted NPS across the cases */
  if $|\Theta| > 1$ *and* $Var(Abs(\Theta_{(c=\hat{NPS}_{arrival})} - 7.5)) > 0$ **then**
    /* Sort cases by their predicted NPS-score */
    $\Theta_{ordered} \leftarrow Sort(\Theta, Abs(\Theta_{(c=\hat{NPS}_{arrival})} - 7.5), ascending)$
if $P_{SLA} > 0$ **then**
  /* Prioritise the sorted queue $\Theta_{ordered}$ further by the current waiting time in the queue w */
  /* Get the subset of customers where the waiting time is above the service level minus one day */
  $\Theta_{priority} \subset \Theta_{(w)} \geq (P_{SLA} - 1)$
  /* Sort by waiting time */
  $\Theta_{priority} \leftarrow Sort(\Theta, \Theta_{(w)}, descending)$
  /* Get the remaining customers */
  $\Theta_{remaining} \subset \Theta_{(w)} < (P_{SLA} - 1)$
  /* Concatenate to new queue $\Theta_{ordered}$, with prioritized customers $\Theta_{priority}$, now first in the line, regardless of $P_{discipline}$ */
  $\Theta_{ordered} \leftarrow \Theta_{priority} \cap \Theta_{remaining}$

Markov chain described in Section 2.2.2, the agent ψ is unassigned from the case, and the time since the agent became idle is updated, while the trace η is removed from Θ.

3.1.8 Start delay

Since an activity cannot commence when an agent is off duty, we introduce a start delay, i.e. the time between when an activity was scheduled to begin and when it could actually begin. In the simulation experiments, this start delay represents the time until business hours begin again. The business hours are defined here as Monday to Friday from 08:00 to 18:00. The details of how the start delay is generated are shown in pseudo code in Algorithm 6. We will not introduce an end delay when activities end outside business hours, as the activity duration model from Study 1 already includes these.

3.1.9 Next activity

The next activity algorithm is a subroutine that prepares the input variables needed for simulating the next activity and the duration of the next activity. After this step, the simulated values are appended to the current trace η. The next activity is generated from the Markov transition matrix described in Section 2.2.2. The activity duration is generated from the simulation model defined in Equation (3).

Algorithm 4: Case assignment.

Data: ordered case buffer $\Theta_{ordered}$, pool of agents $\Psi = (\psi_1, \ldots, \psi_k)$
Result: pool of agents Ψ
/* Create a temporary pool for idle agents A */
$A \leftarrow \emptyset$
/* For every agent in the agent pool */
for ψ *in* Ψ **do**
  /* If agent is not assigned to a case */
  if $\psi_{(i)} \neq \emptyset$ **then**
    /* Add agent to idle pool, remove from agent pool */
    $A \leftarrow \psi$
    $\Psi \leftarrow \Psi \setminus \psi$
    /* Sort agents in idle pool by time last active t */
    $A \leftarrow Sort(A, A_{(t)}, ascending)$
end
/* For every agent in the idle pool A */
for ψ *in* A **do**
  /* For every case in the ordered case buffer $\Theta_{ordered}$ */
  for η *in* $\Theta_{ordered}$ **do**
    /* If case is currently unassigned */
    if $\eta_{(r=\emptyset)}$ **then**
      /* If agent is still unassigned */
      if $\psi_{(i=\emptyset)}$ **then**
        /* Update case resource with agent identifier k */
        $\eta_{(r)} \leftarrow \psi_{(k)}$
        /* Assign agent to the case via case identifier i */
        $\psi_{(i)} \leftarrow \eta_{(i)}$
  end
end
/* Append all agents to agent pool again */
$\Psi \leftarrow \Psi \bigcup A$

Algorithm 5: Case activity simulation

Data: Case buffer Θ, pool of agents Ψ, event log L, current time window start z
Result: event log L, case buffer Θ, pool of agents Ψ

```
/* For each agent                                                          */
for ψ in Ψ do
    /* Assigned to a case                                                  */
    if ψ_(i) ≠ ∅ then
        /* Get the active case of the agent                               */
        η ← Θ_(η|i=ψ_(i))
        /* Get finish-time of current activity, and next time step k      */
        y ← max(η_(t)), k ← max(η_(j)) + 1
        /* While y is still within the current 15-minute window           */
        while y < z + 0.010416 do
            /* Execute Alg. 6 and add a conditional start-delay           */
            v = StartDelay(y)
            /* Execute Alg. 7 to generate next activity and timestamp     */
            η, y ← NextActivity(η, k, v, φ_act, φ_dur, π_dur)
            /* If absorption state is reached: unassign agent, update last activity time
               and remove case from Θ                                     */
            if a_(j+1) = END then
                ψ_(i) ← ∅
                ψ_(t) ← max(η_(t))
                Θ ← Θ \ η
            /* Append activity j (and its attributes) to event log L, and its appropriate
               trace in case buffer Θ                                     */
            L ← L ⋃ η_(j=k)
            Θ_(i=η_(i)) ← η_(j=k)
            /* Update agent pool Ψ with new information for agent ψ       */
            Ψ ← Update(Ψ, ψ)
        end
end
```

Algorithm 6: Start delay subroutine

Data: Finish time for last activity y
Result: Delay in days v
/* Define the total delay */
$v = 0$
/* Get weekday: $\{1, 2, ..., 7\} \in \mathbb{N}$ */
$weekday \leftarrow GetDayOfWeek(y)$
/* Get time of day: $\{1, 2, ..., 24\} \in \mathbb{N}$ */
$timeofday \leftarrow GetTimeOfDay(y)$
if $weekday > 5$ **then**
  /* Add weekday delay: days until Monday plus time until 8:00 */
  $v = (8 - (weekday + \frac{24}{timeofday})) + 0.33334$
else if then
  /* If tomorrow is not Saturday: */
  if $weekday < 5$ **then**
    /* If within work week but before business hours, add delay until 8:00 */
    if $timeofday < 8$ **then**
      $v = 0.33334 - \frac{24}{timeofday}$
    /* If within work week but after business hours, add delay until tomorrow */
    if $timeofday > 18$ **then**
      $v = 1 - \frac{24}{timeofday} + 0.33334$
  /* If tomorrow is Saturday: */
  else if then
    /* If within work week but before business hours, add delay until 8:00 */
    if $timeofday < 8$ **then**
      $v = 0.33334 - \frac{24}{timeofday}$
    if $timeofday > 18$ **then**
      $v = (8 - (weekday + \frac{24}{timeofday})) + 0.33334$

Algorithm 7: Next activity subroutine

Data: trace η, next time step k, activity start delay v, conditional activity-sequence model ϕ_{act}, conditional activity duration model ϕ_{dur}, feature mapping function for duration model π_{dur}
Result: Updated trace η, finish time of current activity y
/* Get last activity and timestamp */
$a_j \leftarrow \eta_{(a|j=k-1)}$
$t_j \leftarrow max(\eta_t)$
/* Simulate the next activity from Equation 4 (Markov chain) */
$a_{(j+1)} \leftarrow \phi_{act}(a_{(j+1)}|a_j)$
/* Simulate duration of $a_{(j+1)}$ from Equation 3, using feature mapping π_{dur}, and add start delay v to overall duration $t_{(j+1)}$ */
$X = \pi_{dur}(\eta)$
$t_{(j+1)} = \phi_{dur}(t_{(j+1)}|X) + v$
/* Append new activity and timestamp to trace */
$\eta_{(a)} \leftarrow a_{(j+1)}$
$\eta_{(t)} \leftarrow t_{(j+1)}$
/* Update finish time for current activity */
$y \leftarrow t_{(j+1)}$

3.1.10 Evaluation

We answer Research Question 2.1 by measuring the average waiting time in the queue within each day d of each simulation run l. This is achieved by generating a time-series measuring the average waiting time of all customers currently in the queue at the beginning of day d, illustrated in Equation 10:

$$AvgWaitingTime = \frac{1}{N}\sum_{l=1}^{N}\frac{1}{D}\sum_{d=1}^{D}(d - H_{(q,l,d)}) \tag{10}$$

Let $H_{(q,l,d)}$ represent a vector of the individual arrival times q of all customers waiting in the queue in simulation run l at day d, as defined in Equation 11, and s denote the status of a case:

$$H \subset \{\eta \in \Theta, |\eta_{(s=waiting)}\} \tag{11}$$

In a similar manner, we answer Research Question 2.2 by calculating the average queue length, the average capacity utilisation, and the average proportion of cases closed. The average queue length is calculated using the definition in Equation 12:

$$AvgQueueLength = \frac{1}{N}\sum_{l=1}^{N}\frac{1}{D}\sum_{d=1}^{D}|H_{(i,l,d)}| \tag{12}$$

Where $H_{(i,l,d)}$ represents a list of case identifiers i for each simulation run l at day d. The average capacity utilisation is defined by Equation 13:

$$AvgCapUtilisation = \frac{1}{N}\sum_{l=1}^{N}\frac{1}{D}\sum_{d=1}^{D}\frac{|\Psi_{(j\neq\emptyset,i,d)}|}{|\Psi_{(i,d)}|} \tag{13}$$

Where $\Psi_{(d)}$ represents the full agent pool at day d, and $\Psi_{(i\neq\emptyset,d)}$ the set of agents assigned to case i on day d. Finally, the average proportion of cases closed during each individual simulation run is calculated as:

$$AvgPercentCasesClosed = \frac{1}{N}\sum_{l=1}^{N}\frac{|\Theta_{(s=closed,l)}|}{|\Theta_{(l)}|} \tag{14}$$

To answer Research Question 2.3, we first define the simulated NPS response from Equation 8 as NPS_{resp}, we then restrict the response to the discrete interval of the theoretical net promoter score $[0, ..., 10]$ using the definition in Equation 15, and finally we round the response, resulting in $NPS_{resp} \in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ (Eq. 16).

$$NPS_{resp} = \begin{cases} 0, & x \leq 0 \\ x, & 0 > x < 10 \\ 10, & x \geq 10 \end{cases} \tag{15}$$

$$NPS_{resp} = \lceil NPS_{resp} \rceil \tag{16}$$

To calculate the average company or process-level net promoter scores NPS across all simulation runs, we use the approach proposed by (Reichheld, 2003), where $\%detractors$ denotes the proportion of customers with $NPS_{resp} < 7$ and $\%promoters$ denotes the proportion of customers with $NPS_{resp} > 8$ in the respective simulation run l:

$$AvgNPS = \frac{1}{N}\sum_{l=1}^{N}\%promoters_{(l)} - \%detractors_{(l)} \tag{17}$$

3.2 Results

3.2.1 Convergence behaviour

To assess the convergence of the simulated queuing systems to a steady state, we calculated the average queue length at the beginning (00:00) of each day in the simulation period. Figure 5 shows the convergence behaviour for all conditions without SLA, Figure 6 for all conditions with SLA. In both sets of conditions, the stabilisation of the queuing systems depended crucially on the the number of agents. While we simulated conditions ranged from three to nine agents, only those with at least seven agents reached a steady state st all.

Note that the conditional arrival rate (see Eq. 6) peaked at approximately 180 days. Since the simulated period started at 01/07/2018, this point in time corresponds to the Christmas holidays, where the conditional arrival rate had been highest in the calibration data.

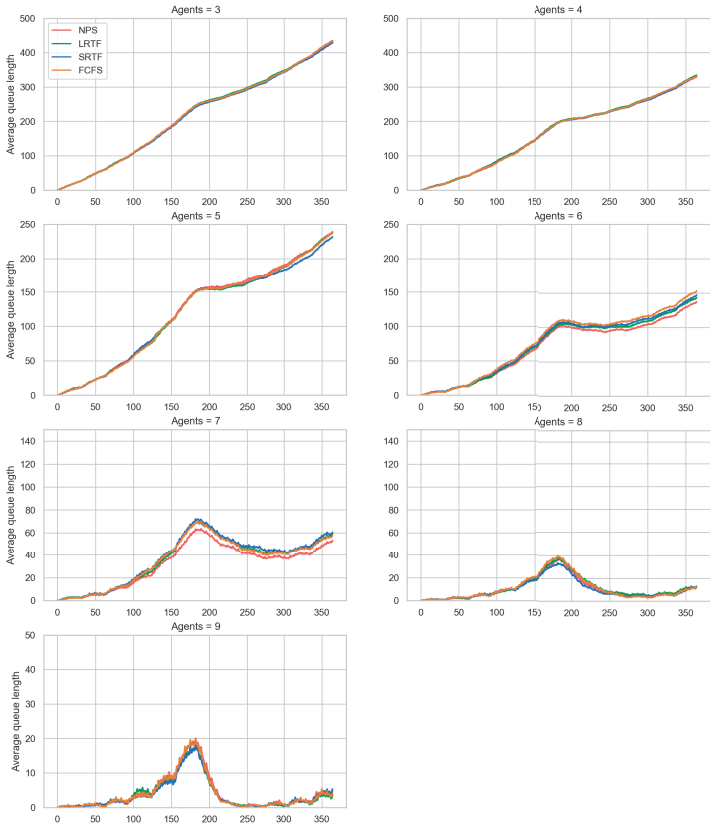


Figure 5: Average daily queue length as a function of simulation day and number of agents (no SLA). **Note:** the range of the y-axis is different for each row of diagrams.

3.2.2 Performance

In a next step, we compared the performance of the four queue disciplines in terms of average queue length, percent capacity utilisation, percent cases closed, average waiting time in the queue, and average case resolution time. We used linear models that included fixed main effects of queue discipline (factor levels: NPS, LRTF, SRTF, FCFS), number of agents (factor levels: 3, 4, 5, 6, 7, 8, 9) and service level agreement (factor levels: none, SLA = 60 hours), plus all

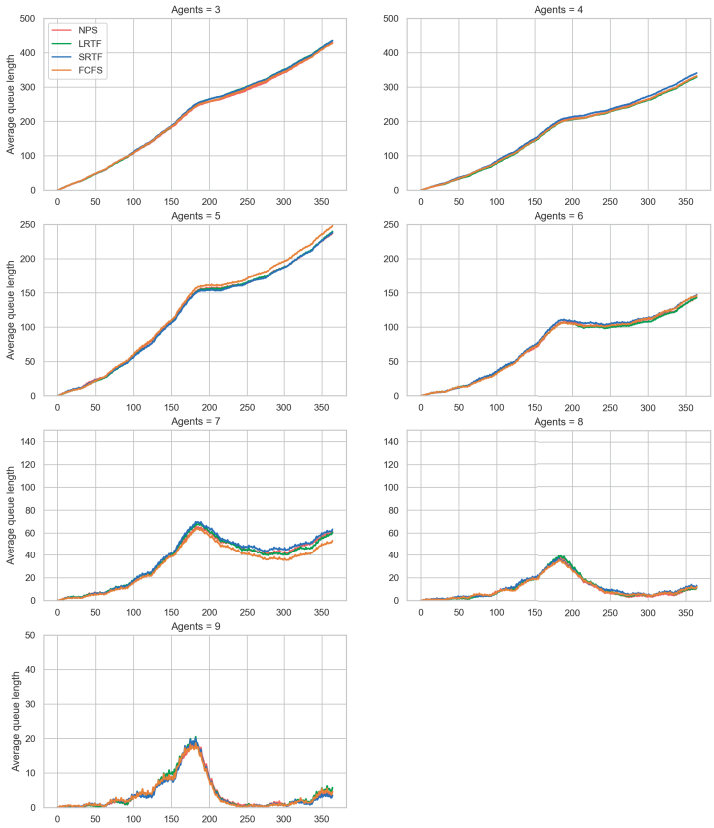


Figure 6: Average daily queue length as a function of simulation day and number of agents (SLA = 60 hours). **Note:** the range of the y-axis is different for each row of diagrams.

two-way interactions and the three-way interaction between the factors. All models were estimated by ordinary least squares.

The number of agents in the scenario had highly significant main effects on average queue length ($F_{6,5544} = 20227.55, p < .001$), percent capacity utilisation ($F_{6,5544} = 7890.69, p < .001$), percent cases closed ($F_{6,5544} = 52078.05, p < .001$), average waiting time in the queue ($F_{6,5544} = 12372.43, p < .001$) and average case resolution time ($F_{6,5544} = 19748.65, p < .001$). Besides the number of agents, there were no other significant effects on average queue length, capacity utilisation, and percent cases closed. This indicates that case throughput and capacity utilisation depended exclusively on the number of agents available for the handling of the incoming cases; neither the queue discipline nor the presence or absence of a service level agreement had additional influences. The means are plotted in Figure 7.

The results were quite different where the duration variables were concerned. Average waiting time in the queue depended on all main effects and interactions in the experimental design, including queue discipline ($F_{3,5544} = 273.80, p < .001$), number of agents ($F_{6,5544} = 12372.43, p < .001$), SLA ($F_{1,5544} = 2442.10, p < .001$), the two-way interaction between queue discipline and number of agents ($F_{18,5544} = 96.03, p < .001$), the two-way interaction between queue discipline and SLA ($F_{3,5544} = 278.80, p < .001$), the two-way interaction between number of agents and SLA ($F_{6,5544} = 828.21, p < .001$), and the three-way interaction between queue discipline, number of agents and SLA ($F_{18,5544} = 97.06, p < .001$).

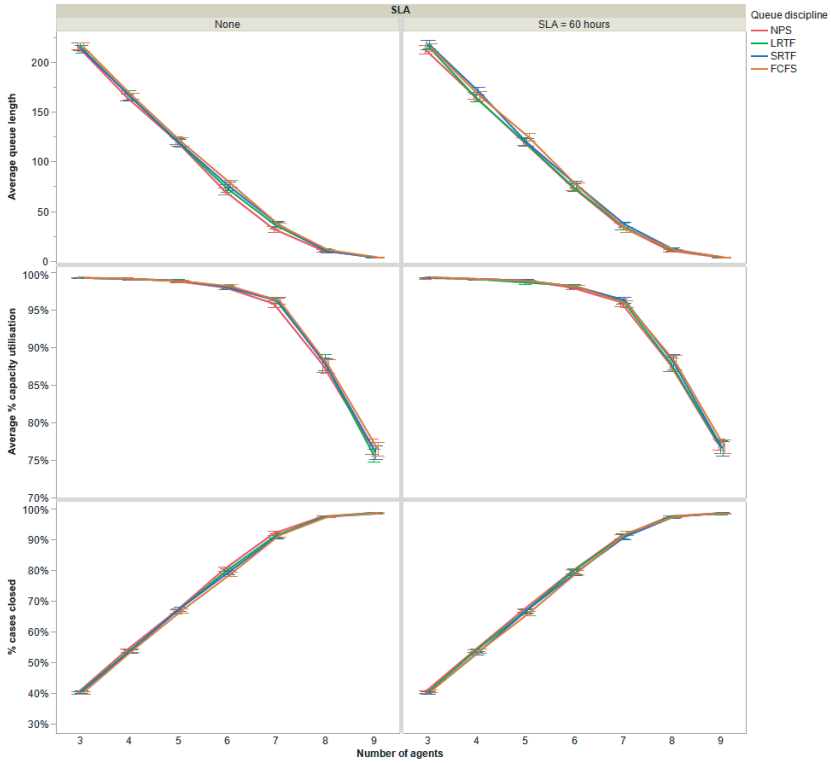


Figure 7: Average queue length, percent capacity utilisation, and percent cases closed as a function of queue discipline, number of agents, and service level agreement (error bars represent 95% confidence intervals).

In a similar manner, average case resolution time depended on all main effects and interactions in the experimental design, including queue discipline ($F_{3,5544} = 3908.00, p < .001$), number of agents ($F_{6,5544} = 19748.65, p < .001$), SLA ($F_{1,5544} = 34552.93, p < .001$), the two-way interaction between queue discipline and number of agents ($F_{18,5544} = 621.18, p < .001$), the two-way interaction between queue discipline and SLA ($F_{3,5544} = 3854.53, p < .001$), the two-way interaction between number of agents and SLA ($F_{6,5544} = 5458.87, p < .001$), and the three-way interaction between queue discipline, number of agents and SLA ($F_{18,5544} = 596.55, p < .001$).

The means are plotted in Figure 8. Compared to FCFS, both NPS and LRTF substantially decreased case resolution time, but did so at the expense of increased waiting time. The performance of NPS and LRTF was identical. SRTF had effects in the same directions but to a lesser degree. All effects disappeared when a service level agreement was present (overriding the respective queue discipline with an FCFS regime) or when enough agents were available so that all arriving cases could be handled without delay.

3.2.3 Effects on customer NPS

In the previous two subsections, we compared the four queuing disciplines in terms of standard convergence and performance metrics. The NPS-based queue discipline developed in this paper showed the same level of performance as its closest "relative" among the competing disciplines, the longest remaining time first (LRTF) discipline. However, the rationale of the NPS-based discipline is to prioritise the cases in the queue in such a way that customer loyalty after case closure is maximised.

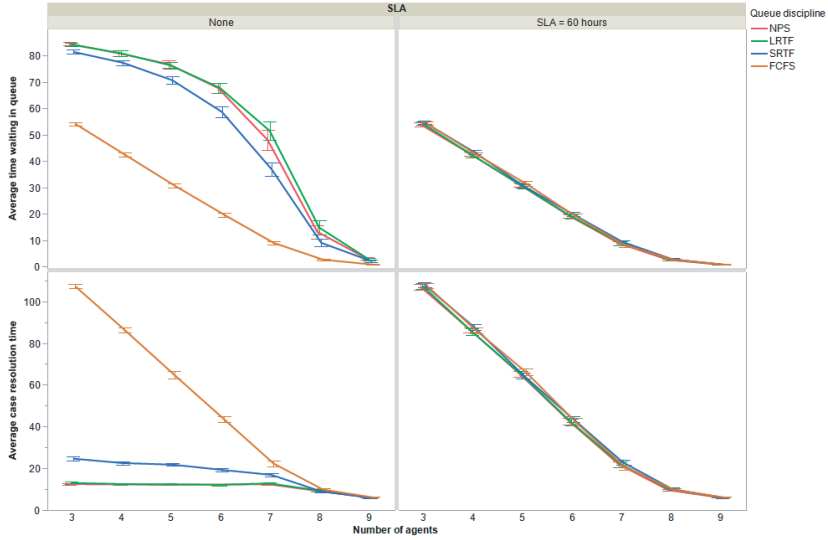


Figure 8: Average time waiting in queue and average case resolution time as a function of queue discipline, number of agents, and service level agreement (error bars represent 95% confidence intervals).

In terms of the average simulated individual NPS responses after case closure, all main effects and interactions in the experimental design were significant: queue discipline ($F_{3,5544} = 27.07, p < .001$), number of agents ($F_{6,5544} = 204.35, p < .001$), SLA ($F_{1,5544} = 243.74, p < .001$), the two-way interaction between queue discipline and number of agents ($F_{18,5544} = 4.02, p < .001$), the two-way interaction between queue discipline and SLA ($F_{3,5544} = 22.09, p < .001$), the two-way interaction between number of agents and SLA ($F_{6,5544} = 37.99, p < .001$), and the three-way interaction between queue discipline, number of agents and SLA ($F_{18,5544} = 2.74, p < .001$). Note that this metric has a minimum of 0 and a maximum of 10; the means are plotted in the left panel of Figure 9.

In terms of the simulated (organisation-level) NPS calculated from the distribution of the individual NPS responses, all main effects and interactions in the experimental design were significant as well: queue discipline ($F_{3,5544} = 20.73, p < .001$), number of agents ($F_{6,5544} = 147.31, p < .001$), SLA ($F_{1,5544} = 187.12, p < .001$), the two-way interaction between queue discipline and number of agents ($F_{18,5544} = 3.04, p < .001$), the two-way interaction between queue discipline and SLA ($F_{3,5544} = 15.78, p < .001$), the two-way interaction between number of agents and SLA ($F_{6,5544} = 31.46, p < .001$), and the three-way interaction between queue discipline, number of agents and SLA ($F_{18,5544} = 3.97, p < .001$). Note that this metric has a minimum of -100 and a maximum of +100; the means are plotted in the right panel of Figure 9.

Compared to FCFS, not only the NPS discipline but also LRTF and SRTF improved both target metrics. The NPS discipline, in turn, was slightly superior to LRTF and SRTF. Again, all effects disappeared when a service level agreement was present (overriding the respective queue discipline with an FCFS regime) or when enough agents were available to handle all arriving cases without delay.

3.2.4 Robustness

The above analyses were performed on the complete period of 365 simulated days. Since all simulation runs had started with an empty queue, there was higher capacity in terms of available service agents during the initial "burn-in" period before the case queue had filled up and the system had reached its steady state. To assess whether the performance differences between the four queue disciplines were robust to the presence or absence of such a burn-in period, we excluded the first 30 simulated days from all simulation runs and repeated all analyses reported above on the resulting 335-day period.

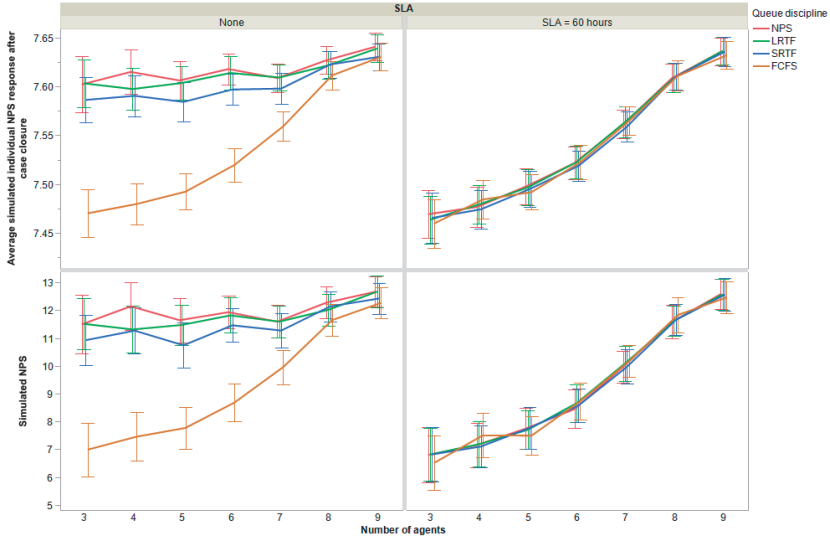


Figure 9: Average simulated individual NPS response after case closure and simulated NPS (on organisation level) as a function of queue discipline, number of agents and service-level agreement (error bars represent 95% confidence intervals).

The results of all statistical tests remained stable, with the exception of the percentage of cases closed. Whilst only the main effect of number of agents had been significant in the analysis of the full 365-day period, all main effects and interactions in the experimental design were significant when we restricted the analysis to the last 335 days: queue discipline ($F_{3,5544} = 12.12, p < .001$), number of agents ($F_{6,5544} = 51652.67, p < .001$), SLA ($F_{1,5544} = 114.52, p < .001$), the two-way interaction between queue discipline and number of agents ($F_{18,5544} = 3.42, p < .001$), the two-way interaction between queue discipline and SLA ($F_{3,5544} = 16.99, p < .001$), the two-way interaction between number of agents and SLA ($F_{6,5544} = 16.88, p < .001$), and the three-way interaction between queue discipline, number of agents and SLA ($F_{18,5544} = 2.88, p < .001$). The means are plotted in Figure 10.

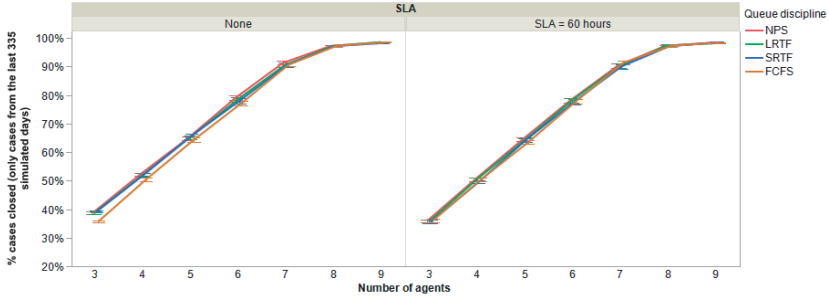


Figure 10: Percent cases closed during the last 335 simulated days as a function of queue discipline, number of agents and service-level agreement (error bars represent 95% confidence intervals).

4 General discussion and conclusion

The aim of the research presented here was to introduce a customer service management approach with a new type of priority queuing based on predicted customer loyalty scores. In our approach, customer loyalty scores are predicted based on predicted case throughput time. Case throughput time is, in turn, predicted from seasonality indicators alone, as case attributes are not available at the time of entering the queue.

We presented two studies to calibrate and test our approach. Study 1 was a statistical analysis of two years of empirical customer service data from a European internet and telecommunications provider. We utilised these real-world data to obtain ecologically valid estimates of the parameters of all distributions governing the customer service process. In Study 2, we used the calibrated distribution parameters in Monte Carlo simulations of the customer service management system, comparing our predictive priority queuing approach with the traditional first come, first served approach (FCFS) and two alternative prioritisation approaches based on predicted throughput time; the longest remaining time first approach (LRTF) and the shortest remaining time first approach (SRTF).

4.1 Key findings

The results indicate that prioritisation based on predicted increases in customer loyalty does indeed lead to an increase in average customer loyalty relative to FCFS, SRTF and LRTF. In addition, it also leads to lower average case throughput time, but only compared to FCSF and SRTF; LRTF performed equally well in this respect as the NPS approach proposed here. Both effects are strong when the number of customer service agents is low but disappear when the number of customer service agents is so high that all cases can be processed immediately. In our opinion, this is the key finding from our Monte Carlo studies. The novel approach we suggested in this paper seems to have most leverage in a service organisation with little excess capacity. The "leaner" the organisation, the more it stands to gain from introducing the prioritisation approach we suggest.

Interestingly, the other two queue disciplines with case prioritisation based on predicted throughput time (i.e., LRTF and SRTF) showed very similar performance patterns as our NPS-based approach. Indeed, the performance of the LRTF discipline was so similar to that of the NPS-based discipline that their difference was not statistically significant on most performance indicators.

Although striking when regarded superficially, the similarity of the behaviour of LRTF and NPS is not really surprising when one considers how cases are prioritised under the two disciplines. Both share an identical first prediction step (predicted throughput time; see Eq. 7). The LRTF discipline prioritises the cases directly based on this prediction. NPS takes the prediction as input to another prediction (NPS; see Eq. 9). However, this second model is linear and does not affect the priority order of the cases. Only the last step in the calculation of the priority rank is non-linear (see Eq. 1), assigning priority ranks to the cases as a function of their proximity to the mid-point of the region defining the "passives" on the scale of individual NPS responses. At least in theory - in practice, however, a large majority of the cases ended up on the same side of that mid-point (because a large majority of the customers in the calibration data had been "promoters" and the distribution was assumed to be the same in the simulation experiments), rendering also the last transformation step predominantly linear and therefore leaving the priority order largely unaffected. In follow-up studies, the distribution of the individual NPS responses should be systematically varied (as opposed to be kept constant and equal to the distribution observed in the calibration data, as was done in the present research) to identify the conditions under which the NPS-based prioritisation discipline will diverge from the LRTF discipline.

Another result that appears striking at first but is not all that surprising when considered more closely, is the disappearance of all throughput time differences between the four queue disciplines when a service level of 60 hours was introduced in order to avoid starvation affects and thereby make the priority-based queue disciplines fairer (see Fig. 8). The way our queuing systems tried to fulfil this constraint was by expediting cases that had been waiting for more than 60 hours to the front of the queue, and if there were several such cases, in the order in which they had entered the queue. Since this way of re-ordering the queue is equivalent to the FCFS discipline, it "overrides" the priority-based disciplines with FCFS. Hence, all performance differences disappeared, at least with the rather stringent service level of 60 hours we applied in Study 2, which had been motivated by the service level agreements currently in use at the case company who had supplied us with the calibration data. In order to reap the benefits of the priority-based queue disciplines investigated here but still live up to elementary levels of fairness, somewhat less stringent service levels should be set. In future research, the service level should be systematically varied in order to identify a level that can be regarded as a reasonable compromise.

4.2 Limitations

In addition to the limitations already discussed above, there were some issues related to the calibration data which we would like to discuss here.

One aspect of Study 1 that might be criticised is the possibility that a systematic selection effect led to the composition of the case and customer sample which we used to train our models and calibrate our simulations. As we described in detail in Section 2.1.1, we only used data from service cases where the customers had actually responded to an SMS request to answer the NPS question. One could argue in the tradition of the exit, voice and loyalty model (Dowding et al., 2000; Hirschman, 1970; Withey and Cooper, 1989) that customers who respond to such requests are on the "voice" side, suggesting that they have a stronger commitment to the relationship with their service provider. However, we do not consider it plausible that the effects in our models would have opposite signs if they were estimated among customers with weaker average commitment. If the effects were weaker but the signs remained the same, our models would still produce similar (albeit noisier) results.

A second limitation is that the calibration data used in Study 1 did not include cases created via phone, as these are not entered into the queue but processed immediately. The prediction models fitted in Study 1 and the simulations reported in Study 2 did therefore not represent this type of case. However, the case company informed us that such cases constitute only a minor proportion of all cases. In addition, we have no information from the case company or any other reason to assume that such cases occur in a systematic manner at specific times such that they would temporarily overload the queue. Hence, we consider our results to be robust against this issue.

A third limitation related to the calibration data in Study 1 is that the case company did not provide information about case abandonment or re-assignment. In the simulations, we therefore assumed the queue prioritisation to be non-preemptive at all times. But even if this assumption was violated, this would merely mean that the mean and the variance of case completion time observed in the calibration data was due to service agents temporarily switching between cases. In Study 2, the bias would have affected the simulations of the four competing queue disciplines in the same way and should therefore not have had an influence on the performance differences *between* the disciplines.

Finally, since the calibration data were obtained from a specific case company in a specific observation period, Study 1 shares the limitations of all case studies: The variation of all features of the service process was limited to the range of variation observed in the calibration data. Hence, the simulations in Study 2 had the great advantage that their parameters *were* based on real-life calibration data, as opposed to assumptions alone. On the other hand, this also had the consequence that the range of scenarios we considered in Study 2 was limited to what appeared reasonable to us in the context of the case company and other companies similar to it. Follow-up studies with systematic variation to factors that has been kept constant here (including variation to more extreme values than used in this case study), will shed additional light on the behaviour of the investigated queue disciplines, and thereby increase the generalisability of our research.

4.3 Applicability

Strictly speaking, the applicability of our approach to service processes in other organisations depends only on one condition: customer loyalty must be inversely related to waiting time. This relationship has to be monotonous, but it does not necessarily have to be strong. In the empirical data we used for calibration (Study 1), the squared multiple correlation between case throughput time and NPS was no more than $R^2 = 0.04$. This is clearly sufficient to obtain strong performance improvements relative to a traditional first come, first served approach (as shown in Study 2). Other empirical studies of the relationship between waiting time and customer satisfaction or loyalty in different types of service settings obtained similar effect sizes (Bielen and Demoulin, 2007; Djelassi et al., 2018; Kumar et al., 1997; Tom and Lucey, 1997). Hence, our approach should normally be applicable in a similar form in other service settings.

Although our approach may in principle be applicable in most customer service settings, its feasibility depends on the availability of case features (e.g., extracted from the CRM system of the organisation) that allow an early prediction of case throughput or waiting time. In the empirical data we used for calibration of our models, no content-related case features were used. The models we used to predict case throughput time (and based on that, NPS gain) only utilised seasonality features - that is, information about the inter-temporal variation of case load and server capacity - but they still performed sufficiently well. If this can be generalised to other service settings, and possibly even improved by including content-related case features if these are available, feasibility should not be a major problem either.

References

R. Bennett and S. Rundle-Thiele. A comparison of attitudinal loyalty measurement approaches. *Journal of Brand Management*, 9(3):193–193, 2002.

F. Bielen and N. Demoulin. Waiting time influence on the satisfaction-loyalty relationship in services. *Managing Service Quality: An International Journal*, 2007.

A. Chaudhuri and M. B. Holbrook. The chain of effects from brand trust and brand affect to brand performance: the role of brand loyalty. *Journal of Marketing*, 65(2):81–93, 2001.

R. M. Dawes. The robust beauty of improper linear models in decision making. *American psychologist*, 34(7):571, 1979.

S. Djelassi, M. F. Diallo, and S. Zielke. How self-service technology experience evaluation affects waiting time and customer satisfaction? a moderated mediation model. *Decision Support Systems*, 111:38–47, 2018.

G. Dobson and A. Sainathan. On the impact of analyzing customer information and prioritizing in a service system. *Decision Support Systems*, 51(4):875–883, 2011.

K. Dowding, P. John, T. Mergoupis, and M. Van Vugt. Exit, voice and loyalty: Analytic and empirical developments. *European Journal of Political Research*, 37(4):469–495, 2000.

I. Gurvich, M. Armony, and A. Mandelbaum. Service-level differentiation in call centers with fully flexible servers. *Management Science*, 54(2):279–294, 2008.

R. Hallowell. The relationships of customer satisfaction, customer loyalty, and profitability: an empirical study. *International journal of service industry management*, 1996.

A. O. Hirschman. *Exit, voice, and loyalty: Responses to decline in firms, organizations, and states*, volume 25. Harvard university press, 1970.

M. K. Hui and D. K. Tse. What to tell consumers in waits of different lengths: An integrative model of service evaluation. *Journal of Marketing*, 60(2):81–90, 1996.

R. Ibrahim. Sharing delay information in service systems: a literature survey. *Queueing Systems*, 89(1):49–79, 2018.

R. Ibrahim and W. Whitt. Wait-time predictors for customer service systems with time-varying demand and capacity. *Operations Research*, 59(5):1106–1118, 2011.

J. Jacoby and R. W. Chestnut. *Brand loyalty: Measurement and management*. Wiley, New York, 1978.

M. U. Kalwani and A. J. Silk. On the reliability and predictive validity of purchase intention measures. *Marketing Science*, 1(3):243–286, 1982.

D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

S. Knox and D. Walker. Measuring and managing brand loyalty. *Journal of Strategic Marketing*, 9(2):111–128, 2001.

S. Knox and D. Walker. Empirical developments in the measurement of involvement, brand loyalty and their relationship in grocery markets. *Journal of Strategic Marketing*, 11(4):271–286, 2003.

P. Kumar, M. U. Kalwani, and M. Dada. The impact of waiting time guarantees on customers' waiting experiences. *Marketing Science*, 16(4):295–314, 1997.

V. Kumar and D. Shah. Building and sustaining profitable customer loyalty for the 21st century. *Journal of retailing*, 80 (4):317–329, 2004.

A. Mahmoumgonbadi, Y. Katebi, and A. Doniavi. A generic two-stage fuzzy inference system for dynamic prioritization of customers. *Expert Systems with Applications*, 131:240–253, 2019.

G. Obermeier, R. Zimmermann, and A. Auinger. The effect of queuing technology on customer experience in physical retail environments. In *International Conference on Human-Computer Interaction*, pages 141–157, Berlin, 2020. Springer.

N. G. Pope. How the time of day affects productivity: Evidence from school schedules. *Review of Economics and Statistics*, 98(1):1–11, 2016.

F. F. Reichheld. The one number you need to grow. *Harvard Business Review*, 81(12):46–55, 2003.

H. A. Reijers. *Design and control of workflow processes: Business process management for the service industry*. Springer, Berlin, 2003.

S. Rundle-Thiele. Exploring loyal qualities: assessing survey-based loyalty measures. *Journal of Services Marketing*, 2005.

S. Rundle-Thiele and M. M. Mackay. Assessing the performance of brand loyalty measures. *Journal of Services Marketing*, 2001.

A. Sayenko, T. Hämäläinen, J. Joutsensalo, and L. Kannisto. Comparison and analysis of the revenue-based adaptive queuing models. *Computer Networks*, 50(8):1040–1058, 2006.

J. A. Schwarz, G. Selinka, and R. Stolletz. Performance analysis of time-dependent queueing systems: Survey and classification. *Omega*, 63:170–189, 2016.

M. Segal. A multiserver system with preemptive priorities. *Operations Research*, 18(2):316–323, 1970.

J. N. Sheth and A. Parvatiyar. The evolution of relationship marketing. *International business review*, 4(4):397–418, 1995.

K. W. Tan, C. Wang, and H. C. Lau. Improving patient flow in emergency department through dynamic priority queue. In *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 125–130. IEEE, 2012.

G. Tom and S. Lucey. A field study investigating the effect of waiting time on customer satisfaction. *The Journal of psychology*, 131(6):655–660, 1997.

W. M. P. Van der Aalst. *Process Mining: Data Science in Action*. Springer, Heidelberg, 2 edition, 2016. ISBN 978-3-662-49850-7. doi: 10.1007/978-3-662-49851-4.

E. Wang, D. Li, B. Dong, H. Zhou, and M. Zhu. Flat and hierarchical system deployment for edge computing systems. *Future Generation Computer Systems*, 105:308–317, 2020.

G. F. Watson, J. T. Beck, C. M. Henderson, and R. W. Palmatier. Building, measuring, and profiting from customer loyalty. *Journal of the Academy of Marketing Science*, 43(6):790–825, 2015.

M. B. Wieth and R. T. Zacks. Time of day effects on problem solving: When the non-optimal is optimal. *Thinking & Reasoning*, 17(4):387–401, 2011.

M. J. Withey and W. H. Cooper. Predicting exit, voice, loyalty, and neglect. *Administrative science quarterly*, pages 521–539, 1989.

Y. Yao, M. Dresner, and K. X. Zhu. "monday effect" on performance variations in supply chain fulfillment: How information technology–enabled procurement may help. *Information Systems Research*, 30(4):1402–1423, 2019.

A Activity duration: Model selection

Table 8: Model comparison: Activity duration (ML: Maximum likelihood, OLS: Ordinary least squares)

Distribution	**Method**	**Validation**	**# Param**	**Train R^2**	**Valid R^2**
Normal	OLS	None	27	0.04475	
Normal	Lasso	Holdback	15	0.0465	0.0373
Normal	Elastic Net	Holdback	27	0.03529	0.05951
Normal	Ridge	Holdback	28	0.01891	0.0121
LogNormal	ML	None	27	0.07969	
LogNormal	Lasso	Holdback	26	0.08224	0.07251
LogNormal	Elastic Net	Holdback	23	0.08420	0.06674
LogNormal	Ridge	Holdback	28	0.07633	0.08477
Weibull	ML	None	27	0.21084	
Weibull	Lasso	Holdback	7	0.17523	0.08078
Weibull	Elastic Net	Holdback	21	0.21347	0.19989
Weibull	Ridge	Holdback	28	0.21048	0.19185

B Activity duration: All model parameters

Table 9: Part 1: Activity duration: All model parameters.

Term	Notes	$\hat{\beta}$	SE	p
Intercept		1.6645	0.1650	<.0001
case_topic[d_2-z_4]		0.0200	0.0456	0.662
case_topic[g_1-z_4]		-0.0538	0.0313	0.0857
case_topic[j_1-z_4]		-0.0557	0.0213	0.0088
case_topic[q_3-z_4]		0.1712	0.0587	0.0035
case_topic[r_2-z_4]		0.0836	0.0383	0.0288
case_topic[w_1-z_4]		-0.0609	0.0286	0.0334
case_topic[w_2-z_4]		0.0119	0.0313	0.7043
case_topic[z_2-z_4]		-0.0420	0.0433	0.3319
case_topic[z_3-z_4]		0.1637	0.0369	<.0001
activity[Email-Task-Reminder]		0.0180	0.1158	0.8767
activity[Interaction-Task-Reminder]		0.1057	0.1319	0.4229
activity_number		0.0420	0.0177	0.0176
resource[a_3-z_4]	Forced in	0.1054	0.0962	0.2731
resource[a_5-z_4]	Forced in	0.0923	0.0831	0.2667
resource[a_6-z_4]	Forced in	0.0852	0.0975	0.3821
resource[a_7-z_4]	Forced in	0.2364	0.0860	0.006
resource[a_8-z_4]	Forced in	0.0508	0.0783	0.517
resource[a_9-z_4]	Forced in	0.1610	0.0843	0.0563
resource[a_10-z_4]	Forced in	0.1152	0.0770	0.1348
resource[b_3-z_4]	Forced in	0.0774	0.0895	0.387
resource[b_4-z_4]	Forced in	-0.0882	0.1011	0.383
resource[b_5-z_4]	Forced in	0.8784	0.3453	0.011
resource[b_6-z_4]	Forced in	0.0893	0.0856	0.2967
resource[b_7-z_4]	Forced in	0.3137	0.1395	0.0245
resource[b_8-z_4]	Forced in	-0.0646	0.0819	0.4304
resource[c_2-z_4]	Forced in	0.2095	0.1011	0.0383
resource[c_3-z_4]	Forced in	0.2009	0.0955	0.0354
resource[d_1-z_4]	Forced in	0.0000	0.0000	1

Table 10: Part 2: Activity duration: All model parameters.

Term	Notes	$\hat{\beta}$	SE	p
resource[e_4-z_4]	Forced in	0.0603	0.0805	0.454
resource[e_5-z_4]	Forced in	0.0896	0.0852	0.2929
resource[e_6-z_4]	Forced in	0.0874	0.0771	0.2566
resource[e_7-z_4]	Forced in	0.1154	0.0833	0.1658
resource[f_2-z_4]	Forced in	0.1227	0.0776	0.1141
resource[f_3-z_4]	Forced in	3.3625	0.2821	<.0001
resource[f_4-z_4]	Forced in	0.4485	0.0820	<.0001
resource[g_9-z_4]	Forced in	-0.0384	0.0836	0.6463
resource[h_4-z_4]	Forced in	0.2234	0.0943	0.0178
resource[h_5-z_4]	Forced in	0.2686	0.1076	0.0126
resource[i_7-z_4]	Forced in	-0.1436	0.0874	0.1002
resource[i_9-z_4]	Forced in	-0.1067	0.0805	0.1851
resource[i_11-z_4]	Forced in	0.1952	0.0913	0.0326
resource[i_12-z_4]	Forced in	0.0271	0.0846	0.7486
resource[j_3-z_4]	Forced in	0.1261	0.0851	0.1383
resource[j_4-z_4]	Forced in	0.0915	0.0793	0.2487
resource[j_5-z_4]	Forced in	0.0673	0.0833	0.4189
resource[j_6-z_4]	Forced in	0.0833	0.0806	0.3011
resource[j_7-z_4]	Forced in	0.0993	0.0940	0.291
resource[j_8-z_4]	Forced in	0.0984	0.0780	0.2071
resource[j_9-z_4]	Forced in	0.0467	0.0774	0.5465
resource[j_10-z_4]	Forced in	0.3716	0.1367	0.0065
resource[l_4-z_4]	Forced in	-0.0284	0.0786	0.7178
resource[l_5-z_4]	Forced in	0.0216	0.0820	0.7923
resource[l_6-z_4]	Forced in	0.7087	0.2148	0.001
resource[l_7-z_4]	Forced in	0.1292	0.0801	0.1068
resource[m_6-z_4]	Forced in	0.0067	0.1011	0.947
resource[m_7-z_4]	Forced in	3.0204	0.3968	<.0001
resource[n_3-z_4]	Forced in	0.1794	0.0841	0.0329
resource[n_4-z_4]	Forced in	0.1639	0.0866	0.0584

Table 11: Part 3: Activity duration: All model parameters.

Term	Notes	$\hat{\beta}$	SE	p
resource[n_5-z_4]	Forced in	0.0328	0.0759	0.6655
resource[p_4-z_4]	Forced in	0.1128	0.0811	0.1644
resource[p_5-z_4]	Forced in	0.1177	0.0781	0.1318
resource[p_6-z_4]	Forced in	0.0702	0.0790	0.3742
resource[p_7-z_4]	Forced in	0.2800	0.0901	0.0019
resource[q_3-z_4]	Forced in	-0.0533	0.1375	0.6986
resource[q_4-z_4]	Forced in	0.0172	0.0844	0.8384
resource[r_2-z_4]	Forced in	0.1459	0.0892	0.1018
resource[r_3-z_4]	Forced in	0.1851	0.0924	0.0452
resource[r_4-z_4]	Forced in	0.0482	0.0880	0.5838
resource[r_5-z_4]	Forced in	0.0932	0.0758	0.2188
resource[r_6-z_4]	Forced in	0.0661	0.0763	0.3857
resource[r_7-z_4]	Forced in	0.0686	0.0761	0.3672
resource[r_9-z_4]	Forced in	0.0918	0.0801	0.2517
resource[s_1-z_4]	Forced in	-0.0159	0.0883	0.8568
resource[s_2-z_4]	Forced in	0.0720	0.0808	0.3725
resource[t_2-z_4]	Forced in	0.4989	0.1339	0.0002
resource[t_3-z_4]	Forced in	0.1595	0.0835	0.0559
resource[t_4-z_4]	Forced in	0.0752	0.0816	0.3566
resource[u_4-z_4]	Forced in	0.0798	0.0853	0.3497
resource[w_6-z_4]	Forced in	0.0901	0.0924	0.3296
resource[w_7-z_4]	Forced in	0.3596	0.1477	0.0149
resource[y_3-z_4]	Forced in	0.0573	0.0922	0.5344
resource[y_4-z_4]	Forced in	0.5338	0.1891	0.0048
resource[y_5-z_4]	Forced in	0.1264	0.0812	0.1197
resource[y_6-z_4]	Forced in	0.2045	0.0903	0.0236
resource[y_7-z_4]	Forced in	0.2302	0.0991	0.0202
resource[y_8-z_4]	Forced in	0.1879	0.0977	0.0545
resource[y_9-z_4]	Forced in	0.0906	0.0869	0.2968
resource[z_3-z_4]	Forced in	0.1188	0.0828	0.1514

C Inter-arrival time: Model selection

Table 12: Model comparison: Arrival rate. ML: Maximum likelihood, OLS: Ordinary least squares.

Distribution	Method	Validation	# Param	Train R^2	Valid R^2
Exponential	ML	None	5	0.3367	
Exponential	Lasso	Holdback	4	0.3823	0.1739
Exponential	Elastic Net	Holdback	5	0.3338	0.3417
Weibull	ML	None	6	0.0668	
Weibull	Lasso	Holdback	5	0.0687	0.0520
Weibull	Elastic Net	Holdback	6	0.0654	0.0525
Normal	OLS	None	6	0.0162	
Normal	Lasso	Holdback	2	0.0000	0.0000
Normal	Elastic Net	Holdback	6	0.0129	0.0206

D Throughput time: Model selection

Table 13: Model comparison: Throughput time (ML: Maximum likelihood, OLS: Ordinary least squares).

Distribution	Method	# Param	Train R^2	Valid R^2
Normal	Adaptive Lasso	4	0.0047	0.0047
Normal	Adaptive Elastic Net	4	0.0047	0.0047
Normal	Lasso	4	0.0046	0.0047
Normal	Elastic Net	7	0.0048	0.0038
Exponential	Adaptive Lasso	5	0.0558	0.0532
Exponential	Adaptive Elastic Net	5	0.0558	0.0532
Exponential	Lasso	6	0.0571	0.0466
Exponential	Elastic Net	3	0.0539	0.0460
Weibull	Adaptive Lasso	6	0.0067	0.0187
Weibull	Adaptive Elastic Net	6	0.0067	0.0186
Weibull	Lasso	7	0.0069	0.0175
Weibull	Elastic Net	7	0.0069	0.0175

E NPS: Model selection (simulation model)

Table 14: Model comparison: NPS simulation model.

Distribution	Method	# Param	Train R^2	Valid R^2
Normal	Lasso	12	0.03527	0.03738
Normal	Elastic Net	12	0.03527	0.03738
Gamma	Lasso	10	0.04059	0.03969
Gamma	Elastic Net	10	0.04059	0.03969
Exponential	Lasso	10	0.00195	0.00210
Exponential	Elastic Net	10	0.00195	0.00210

F NPS: Model selection (prediction model)

Table 15: Model comparison: NPS prediction model.

Distribution	**Method**	**# Param**	**Train R^2**	**Valid R^2**
Normal	Adaptive Elastic Net	3	0.01320	0.03208
Normal	Adaptive Lasso	3	0.01320	0.03208
Normal	Lasso	3	0.01320	0.03208
Exponential	Adaptive Elastic Net	2	0.00074	0.00182
Exponential	Lasso	2	0.00074	0.00182
Exponential	Adaptive Lasso	2	0.00074	0.00182

G Results: NPS interaction effects

Table 16: Least squares means: average actual customer NPS as a function of case prioritisation approach and number of available customer service agents

Factor level combination	Least squares mean	SE
FCFS, 3 agents	8.8872	0.0786
FCFS, 5 agents	8.9669	0.0256
FCFS, 10 agents	9.2428	0.0512
FCFS, 15 agents	9.2499	0.0507
NPS, 3 agents	9.0665	0.0775
NPS, 5 agents	9.1093	0.0254
NPS, 10 agents	9.2421	0.0505
NPS, 15 agents	9.2637	0.0510

H Results: Throughput time interaction effects

Table 17: Maximum likelihood means with logarithmic link function, assuming an exponential distribution: average actual throughput time as a function of case prioritisation approach and number of available customer service agents

Priority scheme	Number of agents	Estimate	Std Error
FCFS	3	4.6741	0.1000
FCFS	5	4.1828	0.1000
FCFS	10	1.6644	0.1000
FCFS	15	1.4705	0.1000
NPS	3	2.5700	0.1000
NPS	5	2.5488	0.1000
NPS	10	1.6626	0.1000
NPS	15	1.4643	0.1000

I Results: Cases closed interaction effects

Table 18: Least squares: Priority scheme vs. Number of agents. F-test P. = 0.6366

Level	Least Sq Mean	Std Error
FCFS, 3	291.05000	2.46556
FCFS, 5	483.19000	2.46556
FCFS, 10	720.26000	2.46556
FCFS, 15	715.69000	2.46556
NPS, 3	292.07000	2.46556
NPS, 5	482.37000	2.46556
NPS, 10	717.45000	2.46556
NPS, 15	719.04000	2.46556

Paper IV

Riess, M. (2023b). Automating model management: A survey on metaheuristics for concept-drift adaptation. Revised version of paper published in Journal of Data, Information and Management (2022), Vol. 4, 211–229.

Automating model management: A survey on metaheuristics for concept-drift adaptation

Mike Riess

School of Economics and Business, Norwegian University of Life Sciences, Universitetstunet 3, Ås, 1433, Norway.

Abstract

This study provides an overview of the literature on automated adaptation of machine learning models via metaheuristics, in settings with concept drift. Drift-adaptation of machine learning models presents a high-dimensional optimisation problem; hence, stochastic optimisation via metaheuristics has been a popular choice for finding semi-optimal solutions with low computational costs. Traditionally, automated concept drift adaptation has mainly been studied in the literature on data stream mining; however, as data drift is prevalent in many areas, analogous solutions have been proposed in other fields. Comparing the conceptual solutions across multiple fields is thereby helpful for the overall progress in this area. The found literature is qualitatively classified in terms of concept drift type and pattern, adaptation approach and type of metaheuristic. It is found that population-based metaheuristics are by far the most widely used optimisation methods across the domains in the retrieved literature. Methodological problems such as evaluation method and transparency in terms of concept drift type tested in the experiments are found and discussed. Over a ten-year period, the usage of metaheuristics in the found literature transitioned from automating single tasks in model development to full automation in recent years. More transparency in terms of evaluation method and data characteristics is important for future comparison of solutions across drift types and patterns. Furthermore, it is proposed that future studies in this area evaluate multiple metaheuristics in each study, in order to illuminate their performance differences in drift adaptation problems.

Keywords: Metaheuristics, Concept drift, AutoML, ML-lifecycle management

1 Introduction

Concept drift is a naturally occurring phenomenon, observed in many different fields [1, 2], such as security and police, financial services, telecommunications, marketing, retail, production, media and others. Concept drift refers to the change in distributions and relationships within the data [3]. When drift occurs, a machine learning model cannot project the previously learnt relationships to the new reality, which leads to degrading predictive performance. Depending on the field of application, the consequences can in some cases be severe [1]. As discussed in [4], models in production (providing predictions to end-users) will in these situations have to be re-trained using data from the new distribution. The amount of effort needed to reach previous performance levels might vary based on drift type, magnitude, and pattern, but is generally unpredictable. Re-training or re-developing machine learning models is in many cases performed manually by professional workers with high salaries and limited capacity [5, 6]. From a business perspective, this can present a trade-off between maintenance and the development of new models. Full or partial automation of model maintenance is thereby more sustainable from a resource utilisation perspective.

Tasks within automated model development [7] and maintenance [8] generally consist of highly complex combinatorial optimization problems, where each step requires solving another computationally demanding optimization problem (called model training). Using exact methods is thereby either directly intractable or too costly. In this case, a group of algorithms called metaheuristics can be particularly useful, as they do not rely on assumptions about the problem structure, nor require perfect information [9]. These methods do not guarantee to find a globally optimal solution, but rather aim to find a semi-optimal solution with minimal effort (being based on heuristics).

The use of metaheuristics is widespread across many application areas, such as business [10], engineering [11], data stream mining [8] and automated machine learning [7], amongst others. However, as the fields using metaheuristics for the adaptation of machine learning models do not necessarily communicate, knowledge and findings might be fractured. A general overview of the literature across the fields will therefore be beneficial in highlighting potential challenges in the area. Comparing the literature in terms of which optimization problem the metaheuristic aim to solve, what type of metaheuristic is used, which machine learning model is adapted, which type of concept drift is studied, and how the proposed solution is evaluated, might therefore help future research.

The contribution of this paper is thereby to: 1) Help understand the general usage of metaheuristics within the literature on self-adapting machine learning models, 2) Classify the use cases in terms of how the metaheuristic assists in self-adaptability, 3) Compare the used methodology of performance evaluation in different settings of concept drift, and finally, 4) Highlight challenges and recommend future directions of research using metaheuristics for drift-adaptation.

1.1 Research questions

Motivated by enhancing the understanding of the usage of metaheuristics for drift-adaptation across multiple fields, this study will retrieve relevant literature and analyse the suggested usage of metaheuristics, as well as their approach to evaluating the proposed algorithms, use cases and/or frameworks. Finally, development of trends in the found literature over time will be studied. To guide this literature review, a set of five research questions are proposed:

- ***RQ*** *How can metaheuristics aid a machine learning system in automatically adapting in settings with concept drift?*

To help answer this primary research question, the following six secondary research questions are proposed:

- ***RQ1*** *Which types of metaheuristics have been utilized for automated adaptation to concept drift?*
- ***RQ2*** *What characterize the application area of the use-cases?*
- ***RQ3*** *How does the use-cases utilize metaheuristics for concept drift adaptation?*
- ***RQ4*** *Which forms of concept drift were investigated?*
- ***RQ5*** *How was the proposed use-cases evaluated?*
- ***RQ6*** *What are the chronological trends in the found literature?*

The purpose of **RQ1** is to get an overview of the application of various metaheuristic algorithms within concept-drift related research. **RQ2** aim at getting an overview of the application areas or overall context of the use-cases in relation to machine learning theory. **RQ3** investigate how the metaheuristic algorithms was applied to help a machine learning system adapt to concept drift. **RQ4** investigate which types of concept drift the use-case was evaluated on, and **RQ5** looks closer at the method and metrics used for evaluation of the proposed methods. Finally, to answer **RQ6** an analysis of the temporal patterns in the found literature is performed to understand the overall development in the area.

2 Background

In the following, fundamental concepts relevant to the literature reviewed in this study will be introduced. The following sections will serve as an overview of the most important related concepts and areas. Terminology introduced in this section will be used in the analysis and discussion of the found literature.

2.1 Machine learning

Machine learning is a sub field of Artificial Intelligence that focuses on developing software that *learns* to perform a task, rather than being hand-coded

by the developer [12]. There are in general 4 different areas of machine learning, each with their own subfields: Supervised learning, Unsupervised learning, Self-supervised learning and Reinforcement learning [13].

In this literature review, the focus is mainly on supervised learning, which can be defined as learning a representation or parameters β in some function $\hat{y} = f(x, \beta)$, which, given some input x will *predict* some output $\hat{y}$. In this case, the parameters are found using a machine learning algorithm, and the resulting *model* can thereby be defined as the function $f()$ with associated parameters β. However, many *non-parametric* machine learning models also exist (such as *random forests* and *k-nearest neighbors* [14]). In the following, the project phases of machine learning model development will be described with focus on the tasks involved with model development.

2.2 Machine learning model development

There exist several normative frameworks for structuring machine learning projects, such as: Cross industry standard process for data mining (CRISP-DM) [15], Sample, Explore, Modify, Model, and Assess (SEMMA) [16] and Knowledge discovery in databases (KDD) [17]. A conceptual overview has been made in [18], which does not conclude that one framework is necessarily superior to the other. The CRISP-DM framework does, however, include *business understanding* in the initial phase of the project, which help align problem and solution. For this reason, CRISP-DM is used to illustrate the basic workflow of a machine learning project. The project work is usually carried out by one or more specialists, most often referred to as *Data scientists* [6] and *Data engineers* [4]. The six steps of CRISP-DM is described briefly in the following.

Step 1: Business understanding. The first step in the CRISP-DM framework is concerned with understanding the requirements and the underlying problem from the business perspective. This insight is thus used as guidance for a machine learning model that help solve the business problem. This phase has also been described as the problem definition phase [13].

Step 2: Data understanding. Next, relevant data is collected for initial analysis. The objective here is to look for patterns in order to form hypotheses for further testing via machine learning experiments. As argued in [15], data understanding is closely related to the business understanding, since the project plan cannot be formulated without having some level of knowledge about the data. Typically, this step would involve explorative data analysis and quality testing of one or more datasets that are available.

Step 3: Data preparation. This step is also known as Extract, Transform and Load (ETL)-step. Here, a data engineer and/or data scientist first extract raw data (files, databases etc.) from the source systems, transform it into a format that is usable for the model(s), and load it prior to the modelling phase. This step includes *normalization*, *one-hot encoding*, *aggregation* and other domain-specific transformations [13]. The data preparation also depends

on the desired types of models (e.g. sequential vs. static). Parts of this step is also commonly referred to as feature preparation or feature engineering, which also often include *feature selection*.

Step 4: Modelling. In the modelling phase, one or more models are trained (described in detail later). The data scientist is unlikely to know from the beginning of the project, which combination of model type and hyper parameters that will give the best result. This phase is thus experimental with a more or less systematic structure. It is most common to use grid-search [19], which is all combinations of a set of model settings (also referred to as *hyper-parameters*). Since the business problem is sufficiently understood at this point, it is important to define a performance metric that illustrate the business value of the model [13].

Step 5: Evaluation. This is the final step before deciding on deployment. At this point, multiple model candidates have been trained and a set of candidate models have been found. The goal now is to evaluate the full procedure and investigate whether mistakes have been made, and/or the developed models actually fulfill the business requirements [15]. This can be done using in-depth analysis of the model performance. Once a model has been sufficiently tested evaluated, and found to satisfy he business requirements, it is selected for the next phase: deployment.

Step 6: Deployment. When the best model candidate have been selected, it needs to be implemented in the system or process it was intended for. This means re-creating the full pipeline (transforming the raw data and predicting from transformed data) made in steps 1-5 in a way that enables real time or batch-prediction of new data, once it is available. Model monitoring is also included as part of deployment [15]. In the case that the model performance degrades, the process goes back to step one and continues from there, forming an infinite loop.

From a process-centric point of view, the model development process can be illustrated as Business Process Model Notation (BPMN) [20] as shown in Figure 1. In this figure, the *monitoring* aspects of the life-cycle have been included, illustrating that once a model is deployed, it is continuously monitored to assess if further development (maintenance) is needed. This is to ensure satisfactory performance, which will be further motivated in sections 2.5 and 2.4.

2.2.1 Batch learning

The vast majority of Machine learning literature is focused on what is referred to as batch learning or offline learning [21]. Here, the assumption is that the distribution of the data is stationary over time, and samples are i.i.d. This means that a classifier or regression model can be trained via partitioning methods such as k-fold cross validation [14], without a need to account for time. With regards to steps 4 and 5 in the CRISP-DM framework illustrated earlier [15], the task of the data scientist is to make decisions on so-called hyper

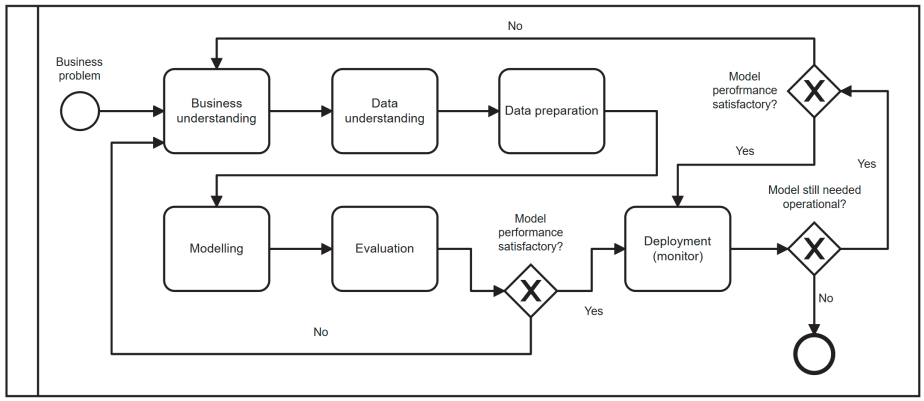


Fig. 1 *BPMN model of the CRISP-DM framework, including monitoring stage.*

parameters and evaluation protocol before selecting a final model candidate for deployment (step 6). An example is provided in the following.

2.2.2 Stochastic gradient descent

A well-known example of a simple yet powerful machine learning algorithm is the stochastic gradient descent algorithm, here used to find the optimal weights of a logistic regression model. The logistic regression model is a linear model with a nonlinear (sigmoid) activation (also called link) function [12]:

$$g(z) = \frac{1}{1 + exp(-z)} \tag{1}$$

The model can thus be defined as:

$$\hat{y} = g(\beta^T X) \tag{2}$$

Where β is a vector of *learnt* weights. For this problem, the optimal weights cannot be found analytically in some cases [22], and a local search method such as SGD is therefore commonly used. A single prediction $\hat{y_i}$ can after learning the optimal weights be made from the linear combination of the weights and the inputs: $\hat{y_i} = g(\beta^T X_i)$. The weights can be learnt by minimizing a loss function, here illustrated by the binary cross entropy loss for classification problems:

$$L = -\sum_{i=1}^{m} (y_i log(\hat{y_i} + (1 + y_i) log(1 - \hat{y_i})) \tag{3}$$

The learning process consist of dividing the available data into multiple batches for out-of-sample validation of the model, most often performed by 3-fold cross validation [14]. The model is trained on a subset of the data called the training set X_{train}, by iteratively adjusting each weight β_j with respect

to the gradient of the loss function for each training example:

$$\frac{\partial}{\partial \beta_j} = L(\beta \mid y, X) = (y - \hat{y})X_j \tag{4}$$

Each update t to β are thus made based on the following update-rule:

$$\beta^{(t+1)} = \beta^{(t+1)} + \lambda(y - \hat{y})X \tag{5}$$

Here, λ is a real-valued scalar between 0 and 1, called the learning rate. The learning rate is a so-called *hyper-parameter* controlling the magnitude in which the weights of β are updated with respect to the gradient of the loss function L. Training the model using Stochastic Gradient Descent (SGD) is illustrated in algorithm 1.

Algorithm 1 Stochastic gradient descent (logistic regression)

```
Initialize weight vector β ≈ U(−1, 1)
for i = 1, ..., n do
    i ← SelectAtRandom[1, n]
    Ŷ_i = g(X_i^T β)
    for Each nonzero feature j of X_ij do
        β_j = β_j + λ(y_i − ŷ_i)x_i
    end for
end for
Return β
```

The SGD-algorithm exists in a variety of forms: another version is the second-order learning algorithm also known as Newton's method [12]. This is known to lead to faster convergence, but is more computationally expensive, as it also requires the calculation of second-order derivatives.

2.2.3 Model evaluation

Performance of the model during training is most often evaluated on a second fold of the data called the validation set X_{valid} using an evaluation metric. As mentioned earlier, this metric has to be in alignment with the business problem, so that the model learns to make predictions that are of business value [13]. An example of a metric for evaluating classification models is the accuracy measure:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{6}$$

This metric can be biased, depending on the balance of the target class; if

only 10 percent of instances in the validation set belong to the negative class, the model could achieve 90 percent accuracy by classifying all instances as positive. To help alleviate these problems, other metrics such as precision, recall and the F1-score is often used:

$$Precision = \frac{(TP)}{(TP + FP)}, Recall = \frac{(TP)}{(TP + FN)} \tag{7}$$

$$F1 = 2\frac{(Precision \times Recall)}{(Precision + Recall)} \tag{8}$$

The F1-score has the advantage that it is controlling for the balance of the target classes. The out-of-sample evaluation can take place during training, and thus out-of-sample performance can be monitored during the training procedure. Corrections to the hyper-parameters is only based on the performance on X_{valid} to avoid *overfitting* [14] to X_{train}. The final model selection is performed using a third and unseen fold called the test set X_{test}.

2.3 Automated machine learning

Also known as *AutoML*, the field of automated machine learning focuses on automating as much as possible of the manual work of the data scientist with regards to the steps of CRISP-DM framework [15]. The field of AutoML has multiple sub-streams of literature such as Meta-learning [23], Neural Architecture Search (NAS) [24], Hyper-parameter optimization (HPO) and Full-model selection (FMS). As this section is not meant to cover all AutoML methods, only HPO and FMS will be considered in the following. Automating parts of machine learning is arguably not a new problem [25], however, it has recently gained much popularity as machine learning has seen a boost in industry adoption, due to increased performance of algorithms and hardware [12]. A common problem across all machine learning projects is the combination of decisions the data scientist has to make in steps 3, 4 and 5 of CRISP-DM, based on steps 1 and 2. These decisions have a direct impact on the level of success, with respect to the performance of the models. A standard heuristic is to select an initial set of candidate settings across steps 3 and 4, train the model(s) on these settings, and evaluate the performance on validation set [19]. The best performing settings are thereafter explored further, depending on the quality requirements and time available to the project team. The main problem of this approach is that it is time-consuming to find the best set of settings, and AutoML can thereby be of help in these cases [7].

2.3.1 Hyper-parameter optimization problem (HPO)

Adapting the definition from [7]; given a machine learning model M, a set of N hyperparameters (learning rate, number of iterations, etc.) with impact on the final solution can be defined. Each *n'th* hyperparameter can

be defined as Λ_n, which is part of the overall hyperparameter configuration space $\Lambda = \Lambda_1 \times \Lambda_2 \times ...\Lambda_N$. A given set of hyperparameters λ for the model is denoted as M_λ. The problem is thereby to find the set of hyperparameters λ^* that minimize the loss over the validation data D_{valid}, given a particular evaluation method: K-fold cross validation or a holdout sample (random partitioning or a temporal split).

$$\lambda^* = \min_{\lambda \in \Lambda} \epsilon(D_{train}, D_{valid})\ \mathbf{V}(\ell, M_\lambda, D_{train}, D_{valid}) \tag{9}$$

Here, the second term $\mathbf{V}(\ell, M_\lambda, D_{train}, D_{valid})$ measures the loss (e.g. Cross-entropy) of the model with the specified settings settings M_λ in the training data D_{train}, evaluated from the validation data D_{valid}. Since this is generally defined as a batch-learning problem, the dataset D is finite and the optimization is thus over the expectation of the sample data D.

2.3.2 Blackbox HPO-methods

As mentioned in [26], blackbox HPO methods can be divided into deterministic and stochastic variants. The deterministic variants rely on linear algebra or geometric methods to find a local optimal solution (due to non-convexity of the HPO-problem) [7], and can be thus be re-started at other starting points to improve convergence towards a global optima.

- **Deterministic methods**
  - Gradient descent [27]
  - Levenberg-Marquardt algorithm [28]

Stochastic methods are mainly based on random variables, statistics, or metaheuristics for guiding the search in order to keep it from being trapped in a local minima. Given sufficient trials, random search has a probability of 1 of finding the global minima [26], while also being more efficient than standard grid search (which is restricted to a fixed set of combinations) [29]. All of the local search algorithms depend on one or more hyperparameters of their own, which determine their probability of finding the global optima within k iterations [7].

- **Stochastic methods**
  - Random search [29]
  - Metaheuristics:
    Local search [30]
    Population-based algorithms [31–34]

2.3.3 Model-based HPO methods

Another direction in HPO is multi-fidelity (model-based) methods as described in [7]. Here, the rationale is to optimize the problem using a low-fidelity version of the problem space. This could be via a smaller subset or compressed version

of the data, leading to a faster search process yielding an approximation for a λ which might perform well on the original (full) problem space. Another model-based approach to HPO is predictive termination [35], where another ML model is tasked with predicting the learning curve of the main model, in order to terminate training before *overfitting* occurs.

2.3.4 Full-model selection

The HPO problem was extended to Full-model selection (FMS) in [36], which is also known as CASHO: Combined Algorithm Selection and Hyper-parameter Optimization [7]. In this context the HPO problem is extended by including elements from steps 3 to 5 in the CRISP-DM framework [15] (Data preparation, Modelling and Evaluation). In [36] the authors include the sub-tasks from CRISP-DM, shown in table 1.

Table 1 *Included subtasks from CRISP-DM*

Step number	Sub-task
3: Data preparation	Data transformation
3: Data preparation	Feature selection (FS)
4: Modelling	Classifier selection
4: Modelling	Hyper-parameter selection
5: Evaluation	Model selection

For Full-model selection, a single solution $X_i \in S$ (where S is the total solution space), is represented in [36] by the following equation:

$$\begin{aligned} X_i = \langle & X_{(i,pre)}, y_{(i,1,\ldots,N_{pre})}, X_{(i,N_{fs})}, \\ & y_{(i,1,\ldots,N_{fs})}, X_{(i,sel)}, X_{(i,class)}, y_{(i,1\ldots N_{class})} \rangle \end{aligned} \tag{10}$$

Here, X_i is a n-dimensional vector representing a particular solution i.e. a full model including data preparation, modelling and evaluation. Each element represented by X in the solution are binary vectors specifying the setting for each step, with the exception of $X_{(i,sel)}$, which is a binary 1-dimensional vector specifying whether data transformation should be performed before feature selection. Additional decisions can be included by adding binary vectors to the solution space (at the expense of increased complexity). The y-vectors contain the hyper-parameter settings for each possible combination of settings in their associated X-vectors. In the example provided by [36], $X_{(i,pre)}$ represent k possible pre-processing techniques such as standardization, scaling or normalization. As some hyper-parameters are continuous, the total search space will be infinitely large. As argued in [7], the continuous parts of the search

space can thereby be bounded or discretized to reduce the overall size. In [36] Particle Swarm Optimization (PSO) [37] is used for solution search. PSO is a population-based metaheuristic (further described in section 2.6) that relies on a fitness function F similar the loss function of the Stochastic Gradient Descent algorithm illustrated earlier. In the case of FMS, the data scientist need to select an evaluation metric for F that represents the business value [13].

2.4 Concept drift

. In real-world machine learning applications, the assumption of stationarity in the data stream is often not fulfilled [4] as data changes over time. This situation called concept drift can arise due to multiple factors, depending on the domain area. As explained in [1], sources can be: Adversary activities (in fraud detection), changes of preferences (in recommender systems), population change or simply due to a complex environment. In [38] the authors further stress that concept drift in consumer-related data is not an error in the data, but rather a natural change in customer behavior. In business process-related data, continuous changes in organisational structures, legal regulations, and technological infrastructures are known to lead to concept drift [2, 39]. Concept drift can be divided into two main categories [40]: Real and virtual concept drift. Real concept drift can be described as the relation between the target variable y and associated input variables X change over time. Virtual drift on the other hand relate to changes in the distribution of input features X, without a change in the relationship between y and X [3]. Given a classification model (such as the logistic regression model presented earlier), a class membership prediction can according to Bayesian theory be made using posterior probability of a class [3]. For a given class $y \in c$:

$$p(y \mid X) = \frac{p(y)p(y \mid X)}{p(X)} \tag{11}$$

Where

$$p(X) = \sum_{y=1}^{c} p(y)p(y \mid X) \tag{12}$$

Real concept drift can thus be defined as a situation where the joint distribution of the target class y and the input data X is significantly different at t_0 compared to t_1:

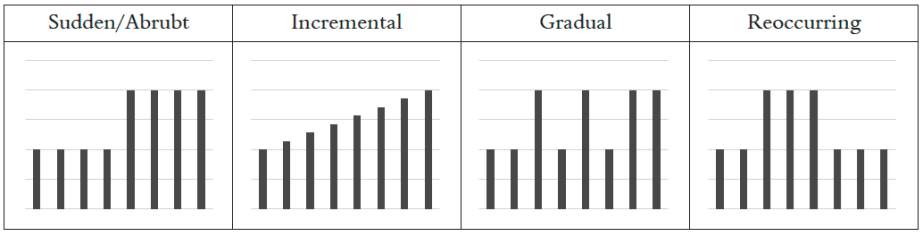
$$\exists X : p_{t0}(X, y) \neq p_{t1}(X, y) \tag{13}$$

Virtual concept drift can be described as situations where the distribution of X changes over time, without changing the decision boundary of y:

$$P_{t0}(X) \neq P_{t1}(X) \wedge P_{t0}(y \mid X) = P_{t1}(y \mid X) \tag{14}$$

The drift between concepts can occur in four different ways, as illustrated in Figure 2. **Sudden or abrupt** concept drift refer to an absolute shift between two concepts as a step between t_0 and t_1. **Incremental** concept drift refer to a steady transition between two concepts at t_0 until t_n, with multiple mixtures of the concepts present in between. At t_n the old concept is non-existing, and a new single concept is dominant in the data. **Gradual** drift refer to situations where there is a back-and-forth change between two concepts happening between t_0 and t_n, where the old concept is non-existing at t_n. **Reoccurring** drift means an introduction of a new concept at time t_{0+n}, with a subsequent move back to the original concept present at t_0. As argued in [3], mixtures of multiple drift-patterns can also be observed in real-world data.

Fig. 2 Overview of drift patterns, adapted from [3]



Feature drift refer to the relevancy over time of each feature in a given feature space $\mathcal{F}$. At a given point in time, the most effective feature set can be selected from the overall feature space $\hat{\mathcal{F}}_{t0} \subseteq \mathcal{F}$. In the case where the most effective feature set changes over time $\hat{\mathcal{F}}_{t0} \neq \hat{\mathcal{F}}_{t1}$, a *feature drift* is present in the data [41]. **Novel class apperance** is a special case of drift where a previously unseen class is observed in the data [42]. In this case $P_{t0}(Y = c) = 0$, and where c is the current unseen class at t_0, and subsequently $P_{t1}(Y = c) > 0$. This can for instance happen in a setting where a model is predicting activities in a business process, and a new type of activity is included in the process. This is also referred to as *concept evolution*.

2.5 Machine learning life-cycle management

Life-cycle management of machine learning projects extend the scope of CRISP-DM to activities performed *post deployment*. Being focused on ML production-settings, *model management* primarily involve the activities listed

in table 2 [43]. As can be seen from the comparison, there is no perfect overlap between model management and CRISP-DM, as business and data understanding is not part of model management, and maintenance is not (explicitly) part of CRISP-DM. Another distinction between the two is that machine learning life-cycle management is focused on the interplay between data engineering and machine learning [4]. Here, the main problem is how to track and utilise metadata across all 6 steps. One of the key challenges in model management is *Maintenance*, where the main issue is when to retrain or re-develop models [4]. This point is determined through performance monitoring, which triggers a new iteration of the model management steps [43]. This is often done offline [4], and often done manually in the interplay between data scientists, software engineers and site reliability engineers [5]. However, many commercial model management systems support (offline) automated HPO [44].

Table 2 Conceptual comparison: CRISP-DM and Model management

Model management steps	**Steps in CRISP-DM**
Data preparation	3: Data preparation
Feature engineering	3: Data preparation
Model training	4: Modelling, 5: Evaluation
Deployment	6: Deployment
Maintenance	

2.5.1 Online learning

Contrary to batch, or *offline* learning, *online* learning is based on the assumption that the distribution of the data is non-stationary [1]. This problem is well-studied within the field of data stream mining [3], where the objective is adaptation of (often streaming-optimised) machine learning algorithms [21]. Online learning aims at automating parts of the CRISP-DM framework [1], in order to adapt the model to changes in the data.

2.5.2 Drift adaptation

Important to online learning is the distinction between *blind* and *informed* adaptation. Blind drift adaptation involves retraining the model when new data appear, or at fixed time intervals [3] without any detection mechanism. This approach can be resource demanding and potentially unnecessary. The idea of *informed* drift adaptation is thereby to detect concept when drift has occurred, and only then adapt the model to the new concept. A framework for drift adaptation is presented in [3]:

1. **Predict:** Predict an incoming sample/batch

2. **Diagnose:** Evaluate using the ground truth once it is available
3. **Update:** Use the new data to update model if needed

The main objectives of drift adaptation is thus to **a)** Detect concept drift as early as possible **b)** Adapt to concept changes while ignoring noise **c)** Perform the operation in less than the time it takes for a new example or batch to arrive, given a fixed budget of memory and computation [1].

2.5.3 Learning modes

Learning modes for concept drift adaptation can be divided into three general forms:*re-training*, *incremental adaptation* and *streaming* [3]. The re-training learning mode discards the existing model and re-train a new one from scratch, based on either old and new, or only new data samples. This approach is the equivalent of batch learning using a sliding window. Incremental adaptation updates the existing model instead of starting from scratch. This learning mode can update the model using either single or multiple samples, once the ground truth (true value of y) have been revealed to the learning algorithm. Finally, the streaming mode is used in settings with a high frequency of incoming samples. In this learning mode, the algorithm uses only few passes over each sample before they are discarded, in order to preserve memory [3].

2.5.4 Drift detection

Since blind adaptation is possible, drift detection is not a necessity for concept drift adaptation. However, there are multiple advantages such as reducing computational load, as well potential insight into the nature of drift in the given setting [3]. Mechanisms for drift detection have 4 main categories: *Sequential analysis* [45, 46], *Control charts* [47], *Distributional tests* [48] and *context-dependent methods* [49]. The main goal of these methods is to monitor the distribution and either alert or trigger automated adaptation (based on the learning mode). A more complete overview of drift detection methods can be seen in [3].

2.6 Metaheuristics

For complex optimization problems as HPO or FMS/CASHO outlined earlier, finding the optimal point in a search space will be computationally demanding. Due to the curse of dimensionality, the volume of the search space grows exponentially with each added dimension. This has the unfortunate sideeffect of an exponential increase in the computation time needed [50]. However, in some cases there exist a set of solutions that are not globally optimal, but "good enough" to solve the problem at hand. In these cases, a particular class of optimization algorithms called Metaheuristics can be useful [9]. The main motivation for these methods is to reduce solution quality in order to solve the problem with less effort (computation time). This is done by trading off exploration and exploitation using robust mechanics [51]. There are mainly

three types of metaheuristics: *population-based*, *construction-based* and *local search* based methods. Each of these will be described with an example in the following.

2.6.1 Local search methods

The local search variants rely on an initial solution and thereafter seek to improve the solution by moving towards the neighboring solutions in iterations. Simple iterative improvement tends to stop at a local minima and yield unsatisfactory results in combinatorial optimization [51]. Multiple improvements have therefore been proposed to the base algorithm over time. An example of a local search algorithm is Simulated Annealing (SA) illustrated in algorithm 2. This method allows moves towards worse solutions, in order to avoid getting stuck in a local minima. This is effectively a mechanism that tries to make the search more explorative. The algorithm has a temperature parameter T which denote the probability of moving towards a worse solution than the current one. This temperature decreases during the search, making the model less likely to explore (make uphill moves), and more likely to exploit the current area of the search space. The decrease of T does not necessarily have to be monotonic and can, depending on the cooling scheme, also increase during the search [51]. An example being $p = exp\left(\frac{f(s')-f(s)}{T}\right)$. Here, s' is the new solution, and s is the old solution. T is the temperature parameter, where T_0 is the initial temperature, which then changes based on the cooling scheme. $GenerateSolution()$ selects an initial solution by random, and $PickAtRandom()$ selects from the neighboring solutions $N(s)$.

Algorithm 2 Simulated Annealing

```
1:  s ← GenerateSolution()
2:  T ← T_0
3:  while termination not met do
4:      s' ← PickAtRandom(N(s))
5:      if f(s') < f(s) then
6:          s ← s'
7:      else
8:          Accept s' as new solution with probability p(T, s', s)
9:      end if
10:     Update(T)
11: end while
```

2.6.2 Population-based methods

This family of methods are based on creating multiple solutions (referred to as individuals), and in some cases by combining superior alternatives in order to

evolve a better set of solutions in the next population. Due to the many variants in this area, only the general evolutionary computation (EC) algorithm is presented in the following. Evolutionary computation is a family of population-based methods inspired by nature, where Genetic Algorithms (GA) are the most well-known variant due to their inspiration from a Darwinian principle: Survival of the fittest [9]. Contrary to Local search-based algorithms, EC generates multiple solutions (populations) per iteration (referred to as a generation), where randomization (called mutation) and combination (also referred to as crossover) influence the algorithms exploration versus exploitation *trade-off*. An example of EC can be seen from algorithm 3.

Algorithm 3 Evolutionary computation

$P \leftarrow GeneratePopulation()$
$Evaluate(P)$
while termination not met **do**
  $P' \leftarrow Recombine(P)$
  $P'' \leftarrow Mutate(P)$
  $Evaluate(P'')$
  $P \leftarrow Select(P' \cup P)$
end while

2.6.3 Constructive methods

Constructive metaheuristics build solutions by combining components of the solution, until a full satisfactory solution is found. As mentioned in [52], these methods often consist of greedy search, where the best elements are picked at each step. Constructive methods can also combine a constructed solution with successive local search. An example of a constructive algorithm is the Ant Colony Optimization (ACO) [53]. This particular metaheuristic is inspired by the way ants search for food in nature [9]. In general, the ants use a pheromone to mark the route they have taken. Multiple paths might then be searched, and while the pheromone vaporizes over time, the shortest path thus has the strongest presence of pheromone [51]. This behaviour is modeled by artificial agents (ants) that perform greedy search on a graph $G(V, A)$, where the nodes of the graph, V are the components of the solution, and A the connections between the components [9]. An example of the approach can be seen from algorithm 4. $ConstructAntsSolutions()$ is the process wherein the solution is created incrementally by the agents in parallel. For each agent, the probability of going from a node k to successor node l is the probability P_{kl}, which is also an increasing function of π_{kl} and $\rho_{kl}(u)$. Let π_{kl} be the *pheromone* on arc (k, l), and $\rho_{kl}(u)$ the *heuristic* value of arc (k, l). Here, the heuristic value is a greedy estimate of the gain by adding (k, l) to the solution [9]. $EvaporatePheromone()$ decrease the pheromone π_{kl} between arcs, every time

an agent uses this particular component. This prevent the algorithm from getting stuck in a local minima. *DeamonActions*() are global (centralized) actions that are performed across all the agents. These vary between the particular implementations of the algorithm [9]. An example is a local search over one or more of the solutions created by the agents, or adding more pheromone to make the search less explorative [51].

Algorithm 4 Ant colony optimization

```
while termination conditions not met do
    ConstructAntsSolutions()
    EvaporatePheromone()
    DeamonActions()
end while
```

3 Methodology

The overall methodology of this study is based on the *semi-systematic* literature review described in [54], which focuses on generating an overview of a given area. This study thereby uses qualitative analysis to classify the literature in terms of relevant theoretical aspects related to the research questions. The literature search have been performed and documented according to the guidelines in [55]. Section 4.1 present the search strategy, queries and their results. The inclusion and exclusion criteria are described in sections 3.2 and 3.3, and finally the analysis approach is discussed in section 3.5.

3.1 Search design

The literature search have been guided by combining three main topics: metaheuristics, concept drift, and automated or adaptive forms of machine learning. The combination of the three topics have been implemented in the queries listed in Table 3 using Boolean **AND**. Since the last two topics exist in multiple forms with different names, synonyms has been defined in each of the three queries using Boolean **OR**. An incremental search approach has been applied, as the aim has been to include as many relevant studies as possible within the combination of the three main topics mentioned above. The first query was designed for *probing*, whereas the second query was designed to be broader than the initial, and finally, the third query imposed a third **AND** clause to further restrict the results. A stopping criteria was defined as obtaining more than 80 percent qualified, redundant results. The queries were performed using Google Scholar since most relevant publishers, proceedings and journals within the subject areas (Springer, Elsevier, IEEE Explore, ACM digital library) have searchable titles and abstracts available online.

Table 3 Queries used

Query no.	Query
1	Metaheuristic **AND** (Concept drift **OR** "Online learning") **AND** ("AutoML" **OR** "Automated Machine Learning" **OR** "Hyper-parameter optimization" **OR** "CASHO" **OR** "FMS")
2	Metaheuristic **AND** "Concept drift" **AND** ("Online learning" **OR** Adaptation **OR** Adaptive **OR** AutoML)
3	Metaheuristic **AND** "Concept drift" **AND** (adaptation **OR** adaptive OR AutoML **OR** Automation **OR** "Hyper-parameter optimization" **OR** "Online learning" **OR** "Full model selection") **AND** ("Machine learning" **OR** "Data stream mining")

3.2 Inclusion criteria

Only one inclusion criteria was formed for this study, namely that the study in question included all a combination of the three topics of interest: Drift adaptation of a Machine learning model using a Metaheuristic algorithm.

3.3 Exclusion criteria

Four exclusion criteria have been defined to ensure the relevancy of the retrieved literature and contain the reviewing effort. The first criterion was included to ensure the content was available, and an analysis of the approach could thereby be performed. The second criterion ensured only peer-reviewed studies were included as a quality assurance criterion. Similarly, criteria three exclude works published before 2021 with less than two citations. This was primarily added to ensure that the suggested method has had an impact in the scientific communities. However, as discussed in section 6, this resembles a snapshot in time, and thereby limits the external validity of the results in the future. The fourth criterion was added to exclude literature that did not suggest or study an explicit method for drift adaptation of a machine learning model using metaheuristics.

1. **Availability**
   - Studies with inaccessible abstract or full text
2. **Peer-review**
   - Unpublished articles (no DOI or publisher)
   - Blog-posts
3. **Scientific impact**
   - Studies with less than two citations when published before 2021
4. **Papers without a suggested method**
   - Position papers

- Literature reviews
- Textbooks

3.4 Study quality assessment

The included studies were mainly assessed in terms of their internal and external validity. In terms of internal validity, the completeness of information was assessed, in terms of the reported target distributions and choice of performance metrics. The type of concept drift, combined with the out-of-sample evaluation method was also compared in order to assess whether the method was evaluated from data that was not accessible during training/calibration. For the external validity, the completeness of information was again assessed in order to understand whether the experiment could be reproduced or compared to similar studies using the same data or data-generating procedure. Whenever these issues are important for the discussion of the respective studies, they will be mentioned in the text in the results section.

3.5 Analysis and classification of methods

The included literature was qualitatively classified in terms of relevant theoretical aspects presented in Section 2 related to the research questions. In cases where the terms presented in section 2 did not cover all approaches in the studies, open coding was used. An overview of the areas of the research questions and their relation to the codes used can be seen below:

- **Types and general applications of metaheuristics - RQ1, RQ2**
  - **Type of metaheuristic:** Algorithm family, algorithm type (open coding)
  - **Field and application area:** Spam detection, finance, medical (open coding)
  - **Machine learning problem type:** Supervised learning (SL), Reinforcement learning (RL), Semi-supervised learning (SSL), Unsupervised learning (USL)
  - **Automated machine learning (AutoML) problem type:** Feature selection (FS), Hyper-parameter optimization (HPO), Full-model selection (FMS)
- **Drift adaptation application areas - RQ3**
  - **Online learning mode:** Retraining, sliding window, incremental adaptation
  - **Drift adaptation method:** Blind, informed
  - **Drift detection method:** Sequential analysis, control charts, distributional test, context-dependent
- **Test of concept drift adaptability - RQ4**
  - **Drift type:** Real drift, virtual drift, feature drift, novel class appearance
  - **Drift pattern:** Sudden, incremental, gradual, reoccurring

  - **Data source:** Real world, synthetic
  - **Data type:** Email, sensor, social media (open coding)
- **Evaluation methodologies - RQ5**
  - **Evaluation method:** K-fold, partitioning, test period, sliding window
  - **Evaluation metric:** Accuracy, recall, precision (open coding)

4 Results

In the following, the results will be divided into five different sections: search results, types of Metaheuristics in the results, adaptation method, test of concept drift adaptability and chronological trends.

4.1 Search results

A total of three queries were performed on May 27, 2022, in the chronological order in which they are listed in table 3. Looking at table 4, *hits* denote the search results, *retrieved* the number of papers retrieved for closer inspection, *redundant* the qualified results already included in a previous query and finally *included* show the number of papers included in the analysis that origin from the given query. The first query led to 258 initial hits with 25 retrieved studies for further inspection. Unfortunately, none of these results were qualified to be included, due to the exclusion criteria (defined in section 3.3). Query 2 led to 570 hits wherein 38 studies were retrieved for closer inspection, resulting in 17 included studies. Finally, query 3 led to 534 hits, with eight qualified for closer inspection. However, all of these were already included from query 2. The total number of included studies is therefore 17.

Table 4 Search results

Query no.	Hits	Retrieved	Redundant	Included
1	258	25	0	0
2	570	38	0	17
3	534	16	8	0

4.2 Types and general application areas of metaheuristics

As can be seen from table 5, population-based metaheuristics is by far the most widely used method across the found studies, in fact, only two studies used local search or construction-based methods [56, 57]. The most frequently used metaheuristic is particle swarm optimization (PSO) [37], which is applied to both continuous as well as discrete optimization [58, 59]. In two of the use-cases, the metaheuristic is combined with Replicator dynamics [60, 61] in

order to utilize the benefit of both approaches in a FMS problem (see section 2.3.4). The fields of application can mainly be grouped into: Engineering, Computer science and Natural language processing. In [58], the focus is mainly to generate self-adaptable models that perform well while using as little memory as possible so that these can be used in e.g. telemetry hardware. Another example is in [62] where the focus is to have self-adaptable models that can compensate for gradual sensor-malfunction in order to save costs in a gas-detection problem. The most common application area is computer science or intelligent systems: in these studies, multiple datasets from different fields are used while demonstrating the efficiency and adaptability of a given algorithm, as compared to other methods. The datasets used (commonly referred to as 'benchmark' datasets) are often retrieved from the UCI Machine learning repository. In the natural language processing approaches, the aim is to classify textual data, which is known to have a high dimensionality as well as both feature drift and novel classes appearance over time. In [63] the authors use AIS for classification of tweets and other social media data. In [57] and [64], the authors use email data for spam classification and automated email folder allocation, respectively. In addition to field-specific data, most studies uses simulated data to be able to test drift adaptation abilities (see table 8).

4.3 Concept drift adaptation

As seen in table 6, most of the use-cases utilise metaheuristics for assisting in supervised learning problems. In these cases, the ground truth is either assumed to be available instantly after the classification has been made [56, 59, 69], or to be available in a given number of time steps after the classification has been made [60]. In 4 of the studies [58, 63, 65, 66] the learning method is unsupervised, in order for the algorithm to be able to adapt and correct itself without knowing the ground truth. These studies therefore use a clustering-approach (DBSCAN, K-means, DEN-STREAM, CLU-STREAM, KDE) in order to segment the data into clusters based on manually specified criteria. In [66] the authors use information entropy of a cluster, to determine if a data-point belong to the given cluster. In most of these studies, the accuracy of the clusters are evaluated using ground truth after the experiment is performed. In one study, the authors in [56] uses reinforcement learning to find the best policy in any given situation (for electricity market trading).

4.3.1 Models used

In general there are 3 categories of ML models applied across the use-cases: 1) Well-known machine learning algorithms (K-NN, MLP/NN, RNN, NB, DBSCAN), 2) Existing streaming-optimized model ensembles (VFDT [59]) and 3) Proposed new (or modifications of existing) algorithms trained using stochastic optimization (Harmony classifier [68], SFLO-TSCS [62], ACDF [57], J-SLNO [70]). The first category is mainly motivated by the known strengths and weaknesses of the algorithms across the various use-cases. The second

Source	MH type	MH	Application field	Data types
[58]	Population	PSO	Engineering	Electrical grid data
[59]	Population	PSO	Multiple (CS, HC)	Biometric data
[65]	Population	DE, WOA, BIA	Multiple (CS, AI, SS)	Simulation
[60]	Population	PSO (PSO+RD)	Multiple (CS)	Multiple (benchmark)
[56]	Pop, Loc.S.	PSO, TS, SA	Decision support	Simulation
[63]	Population	AIS	NLP	Social media data
[66]	Population	PSO	Multiple (CS)	Multiple (benchmark)
[61]	Population	EC/GA (EC+RD)	Multiple (CS)	Multiple (benchmark)
[67]	Population	EC/GA	Engineering	Simulation
[62]	Population	SFLO	Engineering	Sensor data
[57]	Construction	ACO	NLP	Email data
[68]	Population	HS	Multiple (CS)	Multiple (benchmark)
[64]	Population	EC/GA	NLP	Email data
[69]	Population	PSO (Q.PSO)	Multiple (CS, FIN)	Multiple (benchmark)
[70]	Population	J-SLNO	Engineering	Sensor
[71]	Population	GA	Engineering	Sensor
[72]	Population	GA	Multiple (CS)	Multiple (benchmark)

Table 5 Types and application areas. *MH:Meta-heuristic, Loc.S.: Local search, PSO: Particle swarm optimization, DE: Differential evolution, WOA: Whale optimization algorithm, BIA: Bat inspired algorithm, RD: Replicator dynamics, TS: Tabu-search, SA: Simulated annealing, AIS: Artificial immune system, EC: Evolutionary computation, GA: Genetic algorithm, SFLO: Shuffled frog-leaping optimization, ACO: Ant colony optimization, HS: Harmony search, Q.PSO: Quantum particle swarm optimization, J-SLNO: Jaya algorithm and Sea Lion Optimization, CS: Computer science, HC: Healthcare, AI: Artificial intelligence, SS: Social science, NLP: Natural language processing, FIN: Finance.*

category is motivated by the robustness through adaptability of ensemble algorithms in concept drift settings [3, 61]. The third category is mainly proposed for specific cases where the nature of the metaheuristic presents an advantage. For instance, in [68] the authors compare three different Harmony Classifiers (batch, incremental and improved incremental), where computation time is a main motivation for using the Harmony Classifier. The approach in [57] is an ensemble of a modified decision tree algorithm [14] based on ant-colony optimization (ACO), where the main motivation is to improve the average performance of each decision tree, while maximizing heterogeneity in the ensemble at the same time.

Source	**ML problem**	**ML model(s)**	**AutoML problem**	**MH adaptation**
[58]	USL	KDE, Autoencoder	HPO	Cluster initialization
[59]	SL	VFDT	FS	Window and FS
[65]	USL	K-means, D.str., C.str.	MO	Cluster initialization
[60]	SL	Ensemble (DT)	FMS	M.Mgmt., FS
[56]	RL	MLP	MO	What-if analysis
[63]	USL	DBSCAN	FMS	M.Mgmt., FS, N.DT.
[66]	USL	K-NN	FS	FS
[61]	SL	Ensemble	FMS	FS
[67]	SL	RNN, GRU, LSTM	HPO	Window, FS
[62]	SL	SFLO-TSCS	FS	FS
[57]	SL	ACDF, STR-ACDF (RF)	FMS	FMS
[68]	SL	Harmony classifier	MO	MO
[64]	SL	Naive bayes	FS	FS
[69]	SL	MLP	MO	Retraining
[70]	SL	RNN	FMS	FS, HPO
[71]	SL	SARIMA, SVR, MLP	FMS	FMS
[72]	SL	MLP	FMS	FS, HPO, M.Mgmt

Table 6 ML and Auto-ML aspects. *USL: Unsupervised learning, SL: Supervised learning, RL: Reinforcement learning, D.str: DenStream, C.str: CluStream, DT: Decision tree, MLP: Multi-layer perceptron, RF: Random forest, SARIMA: Seasonal Autoregressive Integrated Moving Average, SVR: Suppoort-vector Regression, MO: Model optimization, HPO: Hyper-parameter optimization, FS: Feature selection, FMS: Full model selection, M.Mgmt: Model management, N.DT: Novelty detection.*

4.3.2 AutoML and drift adaptation

In some studies the application of the metaheuristic is not directly related to automated machine learning (AutoML). In these studies, the metaheuristic is only used for model optimization (training), which is not formally an AutoML problem, but rather a generic machine learning problem. One example is using PSO for finding the optimal weights of a neural network (MLP/NN) as an alternative to back-propagation [69], or finding the optimal value of a decision problem [56]. Looking at the adaptation approach in Table 6 (column 5), most of the use-cases utilize metaheuristics for the feature selection (FS) problem, which is a sub-problem of full model selection (FMS). In some use cases, the FS-problem is extended to determining the size of time series windows [59, 67, 71]. In other cases, the metaheuristic is used for the hyper-parameter optimization problem (HPO) in order to adapt the machine learning model in the event re-training is needed. In [67] the authors utilize low-fidelity HPO [7], where initial model candidates are evaluated in subsets of the data, to decrease the overall training time (via early stopping, see [12]).

A general pattern across the use cases is that there are multiple phases of the suggested approach: an initialization phase, most often followed by a online phase. The initialization phase is a generic offline batch-learning AutoML problem. In the subsequent online phase, either FMS or sub-problems such as FS, model selection (MS), and HPO is performed via a metaheuristic. In [63] the authors use the Artificial immune system (AIS) metaheuristic for FMS and perform both feature selection, model selection and novelty detection using this algorithm.

For the use-cases that can be characterized as FMS [57, 60, 61, 63, 70–72], FMS happens either in the initialization phase (offline learning), or in the subsequent online phase via adaptive lifecycle management, which is both blind [57] and informed [60, 61, 63, 72]. In the cases with informed adaptation, methods such as control charts and distribution testing [63, 68] is mainly performed. The clear advantage of this approach is as discussed earlier, that informed training reduces the computational demand. As Neural network-based models are known to be computationally demanding due to a high number of parameters [12], an interesting finding is that only 2 of 6 Neural Network-based approaches [56, 72] uses informed (trigger-based) adaptation. In this case, the adaptation method uses a given error threshold to update the model.

In two cases, FMS is managed by a combination of a metaheuristic and replicator dynamics (RD) to handle model lifecycle management [60, 61]. One example is the RED-PSO framework [60] which uses RD to select which Decision trees to further update, and which to drop based on the error feedback in an online classification setting. Another example is seen in [57], where the information in the pheromone trails in ACO is used for model management, where newer models have a stronger pheromone trail, and older models are gradually dropped. Most of the use-cases are based on incremental learning with a streaming-setting in mind. However, some of the experimental settings are similar to batch learning, as in [67, 68, 70, 71].

In addition, most of the adaptation techniques are blind, which is not necessarily a drawback in terms of concept drift adaptation, but is known to lead to higher computational costs [3]. The drift detection methods employed vary across the found literature.

Source	Learning mode	Adaptation	Drift detection method
[58]	Incremental	Blind	None
[59]	Retrain	Informed	Control charts
[65]	Sliding window	Informed	Context dependent
[60]	Incremental	Informed	Context dependent
[56]	Incremental	Informed	Context dependent
[63]	Sliding window	Informed	Distributional test
[66]	Sliding window	Blind	None
[61]	Sliding window	Informed	Control charts (EDDM)
[67]	Batch	Blind	None
[62]	Sliding window	Blind	None
[57]	Incremental	Blind	None
[68]	Batch, incremental	Informed	Distributional test
[64]	Incremental	Blind	None
[69]	Sliding window	Blind	None
[70]	Batch	Blind	None
[71]	Batch	Blind	None
[72]	Sliding window	Informed	Context dependent

Table 7 Online learning settings

4.4 Test of concept drift adaptability

Unfortunately, multiple studies include little information regarding the concept drift that the proposed solution is tested against. For the domain-specific use cases, real-world data is used [58, 64, 71] to demonstrate the ability to function in this environment, but the nature of the concept drift in the particular data is less transparent. This pattern is present in multiple studies where real-world data is used, except [62], where the authors also demonstrate the fluctuations of the concept (gas sensor readings) over time.

4.4.1 Drift types

In [63] the authors modify real-world data to model the arrival of novel classes in the target variable. This is achieved by adding classes in subsequent batches in the duration of the experiments. Novelty detection in both feature and input space is tested in [65], [63] and [62] and is most often handled using unsupervised learning or feature selection. A commonality for studies with textual data is the need for adaptive feature selection. This is seen in [63], [57] and [64], where feature drift is a natural phenomenon happening in the real-world data that is investigated: novel features (new words) arrive and other features become less important or disappear. Since none of these studies in detail describe the type and magnitude of the concept drift naturally occurring in the real data, the magnitude of the drift cannot be determined. In [60], [66], [61] and [68] the authors compensate for this problem by using simulated data alongside real-world data. In this way, it is possible to control the various types of concept drift and compare the adaptability of the proposed solution in different scenarios. For instance, in [60] the authors use a rotating hyperplane to simulate real concept drift given a small set of features. This particular type of simulation allows to create environments with different magnitudes of change over time. For the following studies; [64, 69–72] the drift type is not described, which unfortunately limits the external validity of the results.

4.4.2 Evaluation methods

In 8 of the 14 studies with classification problems, the accuracy metric was used as the primary metric for model evaluation. As described in section 2.2.3, the accuracy metric is biased towards the majority class, meaning that performance on minority classes is largely overlooked if accuracy is the only performance metric used. Unfortunately, this is the case in [68], [57], [67] and [61], which means that the results could be biased by the balance of the target variable.

Neither of the aforementioned studies report the balance of the target classes, however, the number of classes of each dataset is reported in [57], [68] and [61]. With this in mind, the results of these studies are mainly interesting in terms of the relative performance of the algorithms within the experimental setting of each study. In [57] all datasets have between 11 and 101 target classes, with less than 5000 observations per dataset. In this case, it is possible that one or more target classes could have no true positives, while the accuracy would still be high (assuming that the target classes were balanced). Fortunately the authors in [59, 62–66, 72] also evaluate the classification performance using either confusion matrix, ROC-index, f1 metric or precision/recall performance, which all take the target balance into account.

Evaluation is most often performed using two-fold partitioning of the data, however, this vary across the studies depending on whether incremental learning or the sliding window approach is used. In [63] novelty detection capability is evaluated using a train and test period, where 2 concepts are present in the

Source	Drift type	Drift pattern	Data source	Evaluation method
[58]	Virtual drift	Unknown	Real world	K-fold
[59]	Real drift	S, G, R	Real world	Test-period (sl. win.)
[65]	Real, novel class	S, G	Synthetic	Test-period (sl. win.)
[60]	Real, feature drift	S, G, R	Synth., Real world	Test-period (sl. win.)
[56]	Real drift	Unknown	Synthetic	Test-period (sl. win.)
[63]	Real, feature, novel c.	Unknown	Real world	Test-period (sl. win.)
[66]	Unknown	Unknown	Synth., Real world	Test-period
[61]	Real, feature drift	S, G, R	Synth., Real world	Test-period (sl. win.)
[67]	Unknown	Unknown	Synthetic	K-fold
[62]	Feature drift, novel c.	S, G	Real world	Test-period
[57]	Feature drift, novel c.	Unknown	Real world	K-fold
[68]	Real, feature drift	G	Synth., Real world	Test-period (sl. win.)
[64]	Unknown	Unknown	Real world	K-fold
[69]	Unknown	S, G	Real world	Test-period (sl. win.)
[70]	Unknown	Unknown	Real world	K-fold
[71]	Unknown	Unknown	Real world	Test-period
[72]	Unknown	Unknown	Real world	K-fold

Table 8 Drift types and evaluation methods. *S: Sudden, G: Gradual, R: Reoccurring, I: Incremental, Sl. Win: Sliding window., Novel c.: Novel class., Synth: Synthetic.*

training period, and 4 concepts in the test period (by adding two novel classes). In [62] the authors use a training period of one batch (initialization phase) and subsequently use nine test batches to evaluate the online performance of the suggested approach.

4.5 Chronological trends

To illustrate some of the trends in the included literature over time, each study have been qualitatively coded into categories related to the type and family

of the used metaheuristics, as well as the type of concept drift and AutoML problem. The results can be seen from Figure 3.

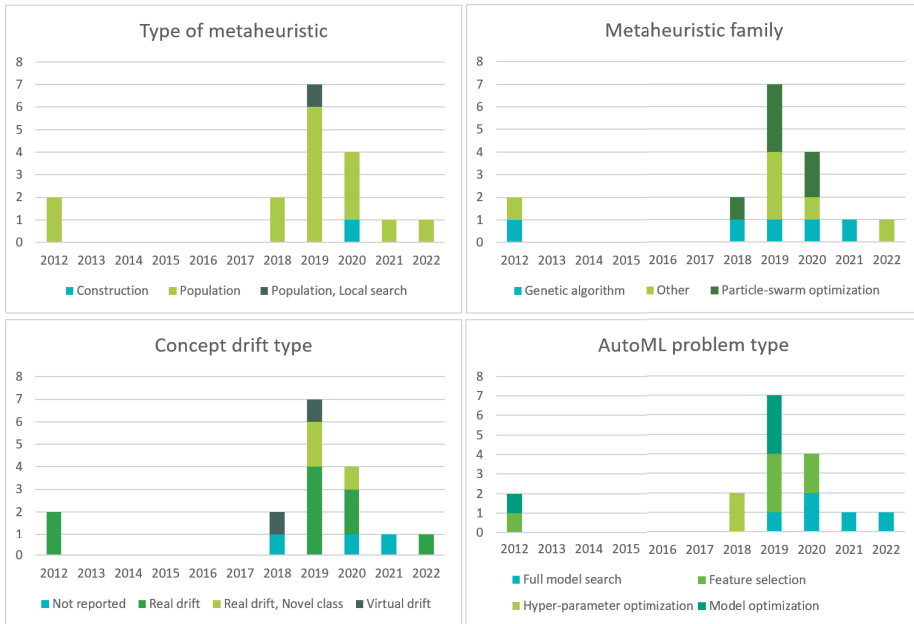


Fig. 3 Chronological trends in found literature. Top-left: type of metaheuristic. Top-right: Family of proposed or used metaheuristic. Lower-left: Drift type included in the study. Lower-right: AutoML problem type.

Looking at the types of metaheuristic, it can be seen that the vast majority of the studies are population-based as mentioned in section 4.2. Only two of the studies use other types of metaheuristics: Construction-based [57] and local search [56]. The proposed metaheuristics can be further analyzed by categorizing which existing algorithm it is derived from, in the cases where it is possible based on its description. Looking at the upper-right diagram in Figure 3, it can be observed that 11 of the proposed algorithms were derived from either Genetic algorithm (GA) or Particle-swarm optimization (PSO). Where PSO has seen the most interest in 2018 to 2020, the GA-based variants are more stable over time.

In terms of the drift types studied, the majority of the studies in the period include real concept drift. In some of the works, the drift type is not reported as mentioned in section 4.4.1, however, there do not seem to be a temporal trend. Finally, an interesting pattern in the type of AutoML problem can be observed by the lower-right diagram in Figure 3, as the early works found from 2012 mainly focus on conventional Machine learning problems: Model optimization and feature selection. This trend generally continues until 2019 where *Full model selection (FMS)* [7] becomes more dominating.

5 Discussion and future work

The initial aim of this study was to map the literature on drift adaptation of machine learning models via metaheuristics. As the results of this literature review show, multiple optimization problems studied in automated machine learning (FS, HPO, FMS) have been addressed and implemented as online versions in the found literature [57, 60, 61].

Based on the retrieved literature, the evaluation of concept drift adaptation in itself can be complex, and the level of details reported seem to vary across studies. Methods for evaluating machine learning model performance, and especially concept drift adaptation, differ across the found studies. Future research should thereby focus on using evaluation metrics that are unbiased with respect to the balance of the target variable (in the case of classification problems).

In addition, as the assumption that the ground truth is available immediately after the prediction might not be valid in many situations, the generalizability of results can be further improved by additionally testing scenarios where the ground truth is delayed, as in [62]. Evaluating the proposed approaches using both real-world data with drift in combination with simulated drift, gives the advantage of both ecological validity, as well as transparency of the tested drift type and pattern. A general downside of the found studies using real-world data alone, is the lack of details on the nature of the concept drift in the data. In future studies, more emphasis should thereby be put on the nature of the drift in real-world data sources.

The results of this study show that multiple solutions exist for combining black-box optimisation [7] with machine learning in order to automate what is referred to as *model maintenance* in ML life-cycle terminology [43]. The analysis of the chronological trends also illustrate that the literature has advanced over the years from solely focusing on single tasks in the CRISP-DM framework [15], to performing online FMS in data streams with concept drift. However, the initial step of aligning the project goals, evaluation metrics and objective functions remains a manual task. Amongst the used metaheuristics, population-based methods were most frequently used. One reason for this could be due to the benefits of parallel computation (training multiple models simultaneously), their simplicity, or individual context-dependent strengths [57, 63].

It remains unclear whether one population-based approach performs better than another in which scenario, as different population-based metaheuristics were not compared to each other in any of the found studies (only variants of the same algorithm such as [60, 61, 68, 69]). Future work might therefore study the effect of drift patterns and type of population-based metaheuristic on the long-term computational cost, etc. Another area that might be investigated is the influence of long-term drift adaptation and performance of business processes.

6 Threats to validity

A general threat to the validity of this study is a potential selection bias in the retrieval of literature. To make this bias as transparent as possible, the literature search, inclusion and exclusion criteria in the study selection process have been documented. One potential source of selection bias is the exclusion of literature with less than two citations (exclusion criteria 3), which could lead to excluding studies that did indeed meet the other criteria. Therefore, the results in this study must be viewed as a snapshot in time, illustrating an overview of the approaches to drift adaptation with the most scientific impact at the time of writing. Another potential source of selection bias is exclusion criteria 2, which excludes blog posts, masters theses and textbooks which might all present a valid method that combines all three topics of this study. Furthermore, as multiple fields in some cases have different terms for the same concept, an interpretation bias from the researcher examining the found studies is also a potential threat to the validity of the results. It is indeed possible that one of the three topics in this study is referred to using non-standard terms in one or more fields, and that important work was thereby not included in this review. Finally, as the algorithm behind the Google Scholar search engine could be updated over time or content could be removed, it is possible that the queries in this study will not reproduce the same search results.

7 Conclusion

The results show that population-based metaheuristics are the most popular methods in the found literature. In particular, Genetic Algorithms and Particle-Swarm Optimization are two metaheuristics frequently used across multiple fields (engineering, computer science, managerial decision support, finance, and social science). As neither of the found studies compares different population-based metaheuristics to each other, it remains unclear whether one variant is superior to another across the problem use cases. It is therefore suggested that future research focus on comparing not only the adaptive machine learning models but performance across metaheuristics as well. The proposed approaches in the found literature are evaluated using either real-world data or a combination of synthetic and real-world data sets. In terms of drift-adaptation, the metaheuristics in the early literature are primarily used to automate single tasks in machine learning development, such as feature selection or hyper-parameter optimization. In more recent literature (at the time of writing), full model selection is a more widespread utilization of metaheuristics for drift adaptation. It is found that some of the retrieved literature is lacking in the reporting of machine learning model performance: In 4 of the 17 retrieved studies, the class distribution of the target variable is not reported while accuracy is the only metric used for evaluating model performance (potentially leading to biased results in data with unbalanced target class distributions). A general problem in the found literature is the lack of

details reported regarding drift type and pattern. In studies with synthetic or a mixture of synthetic and real-world data, the drift type and pattern are often transparent; however, for the studies using real-world data exclusively, these details are often not reported. Future work in this area should therefore include drift characteristics alongside the relative performance of the proposed solutions.

References

[1] Žliobaitė, I., Pechenizkiy, M., Gama, J.: An overview of concept drift applications. (2016)

[2] Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring, pp. 1–8 (2017). https://doi.org/10.1109/SCC.2017.10

[3] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys (2013)

[4] Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., Szarvas, G.: On challenges in machine learning model management. IEEE Data Eng. Bull. **41**, 5–15 (2018)

[5] Polyzotis, N., Roy, S., Whang, S.E., Zinkevich, M.: Data lifecycle challenges in production machine learning: A survey. SIGMOD Rec. **47**(2), 17–28 (2018). https://doi.org/10.1145/3299887.3299891

[6] H., D.T., Patil, D.J.: Data scientist: The sexiest job of the 21st century. Harvard Business Review 90, 70–76 (2012)

[7] Feurer, M., Hutter, F.: In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Hyperparameter Optimization, pp. 3–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_1. https://doi.org/10.1007/978-3-030-05318-5$_1$

[8] Ghomeshi, H., Gaber, M.M., Kovalchuk, Y.: Eacd: Evolutionary adaptation to concept drifts in data streams. Data Mining and Knowledge Discovery **33**(3), 663–694 (2019)

[9] Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J.: A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing (2008)

[10] Hemasian-Etefagh, F., Safi-Esfahani, F.: Dynamic scheduling applying new population grouping of whales meta-heuristic in cloud computing. The Journal of Supercomputing **75**(10), 6386–6450 (2019)

[11] Tomoiagă, B., Chindriş, M., Sumper, A., Sudria-Andreu, A., Villafafila-Robles, R.: Pareto optimal reconfiguration of power distribution systems using a genetic algorithm based on nsga-ii. Energies **6**(3), 1439–1455 (2013)

[12] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, ??? (2016). http://www.deeplearningbook.org

[13] Chollet, F.: Deep Learning with Python. Manning, ??? (2017)

[14] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer, New York, NY, USA (2001)

[15] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., *et al.*: Crisp-dm 1.0: Step-by-step data mining guide. SPSS inc **9**, 13 (2000)

[16] Matignon, R.: Data Mining Using SAS Enterprise Miner. John Wiley & Sons, ??? (2007)

[17] Fayyad, U.M.: Data mining and knowledge discovery: making sense out of data. IEEE Expert **11** (1996)

[18] Shafique, U., Haseeb, Q.: A comparative study of data mining process models (kdd, crisp-dm and semma). International Journal of Innovation and Scientific Research (2014)

[19] Raschka, S., Mirjalili, V.: Python Machine Learning, 3rd Ed., 3rd edn. Packt Publishing, Birmingham, UK (2019)

[20] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., *et al.*: Fundamentals of Business Process Management vol. 2. Springer, ??? (2018)

[21] Barddal, J.P., Gomes, H.M., Enembreck, F., Pfahringer, B.: A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. Journal of Systems and Software **127**, 278–294 (2017). https://doi.org/10.1016/j.jss.2016.07.005

[22] Lipovetsky, S.: Analytical closed-form solution for binary logit regression by categorical predictors. Journal of Applied Statistics, 37–49 (2015)

[23] Khan, I., Zhang, X., Rehman, M., Ali, R.: A literature survey and empirical study of meta-learning for classifier selection. IEEE Access **8**, 10262–10281 (2020)

[24] Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. Journal of Machine Learning Research **20**(55), 1–21 (2019)

[25] Bengio, Y.: Gradient-based optimization of hyperparameters. Neural Computation **12**(8), 1889–1900 (2000). https://doi.org/10.1162/089976600300015187

[26] Muñoz, M.A., Sun, Y., Kirley, M., Halgamuge, S.K.: Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. Information Sciences **317**, 224–245 (2015). https://doi.org/10.1016/j.ins.2015.05.010

[27] Maclaurin, D., Duvenaud, D., Adams, R.P.: Gradient-based hyperparameter optimization through reversible learning. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, pp. 2113–2122. JMLR.org, ??? (2015)

[28] Strijov, V., Weber, G.W.: Nonlinear regression model generation using hyperparameter optimization. Computers and Mathematics with Applications **60**(4), 981–988 (2010). https://doi.org/10.1016/j.camwa.2010.03.021. PCO' 2010

[29] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(null), 281–305 (2012)

[30] Yoo, Y.: Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches. Knowledge-Based Systems **178**, 74–83 (2019). https://doi.org/10.1016/j.knosys.2019.04.019

[31] Bibaeva, V.: Using metaheuristics for hyper-parameter optimization of convolutional neural networks. In: 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6 (2018)

[32] Matuszyk, P., Castillo, R.T., Kottke, D., Spiliopoulou, M.: A comparative study on hyperparameter optimization for recommender systems. In: Lex, E., Kern, R., Felfernig, A., Jack, K., Kowald, D., Lacic, E. (eds.) Workshop on Recommender Systems and Big Data Analytics (RS-BDA'16) @ iKNOW 2016 (2016). http://socialcomputing.know-center.tugraz.at/rs-bda/

[33] Hutter, F., Hoos, H., Leyton-Brown, K.: An evaluation of sequential model-based optimization for expensive blackbox functions. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation. GECCO '13 Companion, pp. 1209–1216. Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2464576.2501592. https://doi.org/10.1145/2464576.2501592

[34] Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi,

F.M., Rizzi, W., Simonetto, L.: Genetic algorithms for hyperparameter optimization in predictive business process monitoring. Inf. Syst. **74**(P1), 67–83 (2018). https://doi.org/10.1016/j.is.2018.01.003

[35] Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the 24th International Conference on Artificial Intelligence. IJCAI'15, pp. 3460–3468. AAAI Press, ??? (2015)

[36] Escalante, H.J., Montes, M., Sucar, L.E.: Particle swarm model selection. Journal of Machine Learning Research **10**(15), 405–440 (2009)

[37] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–19484 (1995)

[38] Akila, S., Reddy, U.S.: Cost-sensitive risk induced bayesian inference bagging (ribib) for credit card fraud detection. Journal of Computational Science **27**, 247–254 (2018). https://doi.org/10.1016/j.jocs.2018.06.009

[39] Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Handling concept drift in process mining. In: Mouratidis, H., Rolland, C. (eds.) Advanced Information Systems Engineering, pp. 391–405. Springer, Berlin, Heidelberg (2011)

[40] Tsymbal, A.: The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Trinity College Dublin, 58 (2004)

[41] Nguyen, H.-L., Woon, Y.-K., Ng, W.K., Wan, L.: Heterogeneous ensemble for feature drifts in data streams. (2012). https://doi.org/10.1007/978-3-642-30220-6_1

[42] Webb, G., Hyde, R., Cao, H., Nguyen, H.-L., Petitjean, F.: Characterizing concept drift. Data Mining and Knowledge Discovery **30** (2015). https://doi.org/10.1007/s10618-015-0448-4

[43] Vartak, M., Madden, S.: Modeldb: Opportunities and challenges in managing machine learning models. IEEE Data Eng. Bull. **41**, 16–25 (2018)

[44] Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., Zumar, C.: Accelerating the machine learning lifecycle with mlflow. IEEE Data Eng. Bull. **41**, 39–45 (2018)

[45] Page, E.S.: Continuous inspection scheme. Biometrika (1954)

[46] Pesaranghader, A., Viktor, H.L.: Fast hoeffding drift detection method for evolving data streams. In: ECML/PKDD (2016)

[47] Ross, G.J., Adams, N.M., Tasoulis, D.K., Hand, D.: Exponentially weighted moving average charts for detecting concept drift. Pattern Recognition Letters 33 (2012)

[48] Bifet, A., Gavaldà, R.: Exponentially weighted moving average charts for detecting concept drift. Proceedings of the Seventh SIAM International Conference on Data Mining (2007)

[49] Bouchachia, H.: Fuzzy classification in dynamic environments. Soft computing (2011)

[50] Chen, S., Montgomery, J., Bolufé-Röhler, A.: Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. Applied Intelligence (2015)

[51] Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003). https://doi.org/10.1145/937503.937505

[52] Trabelsi, K., Sevaux, M., Coussy, P., Rossi, A., Sörensen, K.: Metaheuristics, (2010)

[53] Dorigo, M., Di Caro, G.: Ant colony optimization: A new meta-heuristic, vol. 2, pp. 1477–2 (1999). https://doi.org/10.1109/CEC.1999.782657

[54] Snyder, H.: Literature review as a research methodology: An overview and guidelines. Journal of business research **104**, 333–339 (2019)

[55] Kitchenham, B.: Procedures for performing systematic reviews. Keele, UK, Keele University **33**(2004), 1–26 (2004)

[56] Pinto, T., Vale, Z., Sousa, T., Praça, I., Santos, G., Morais, H.: Adaptive learning in agents behaviour: A framework for electricity markets simulation. Integrated Computer-Aided Engineering **21**, 399–415 (2014). https://doi.org/10.3233/ICA-140477

[57] Kozak, J., Juszczuk, P., Probierz, B.: The hybrid ant colony optimization and ensemble method for solving the data stream e-mail foldering problem. Neural Computing and Applications (2020). https://doi.org/10.1007/s00521-019-04672-1

[58] Bessa, R., Sampaio, G., Miranda, V., Pereira, J.: Probabilistic low-voltage state estimation using analog-search techniques, pp. 1–7 (2018). https://doi.org/10.23919/PSCC.2018.8443074

[59] Lan, K., Fong, S., Liu, L.-s., Wong, R., Dey, N., Millham, R., Wong, K.: A clustering based variable sub-window approach using particle swarm optimisation for biomedical sensor data monitoring. Enterprise Information Systems (2019). https://doi.org/10.1080/17517575.2019.1597388

[60] Ghomeshi, H., Gaber, M., Kovalchuk, Y.: A non-canonical hybrid metaheuristic approach to adaptive data stream classification. Future Generation Computer Systems (2019). https://doi.org/10.1016/j.future.2019.07.067

[61] Ghomeshi, H., Gaber, M., Kovalchuk, Y.: Eacd: evolutionary adaptation to concept drifts in data streams. Data Mining and Knowledge Discovery (2019). https://doi.org/10.1007/s10618-019-00614-6

[62] Rehman, A., Bermak, A., Hamdi, M.: Shuffled frog-leaping and weighted cosine similarity for drift correction in gas sensors. IEEE Sensors Journal **PP**, 1–1 (2019). https://doi.org/10.1109/JSEN.2019.2936602

[63] Abid, A., Jamoussi, S., Ben Hamadou, A.: Ais-clus: A bio-inspired method for textual data stream clustering. Vietnam Journal of Computer Science **6** (2019). https://doi.org/10.1142/S2196888819500143

[64] Cortez, P., Vaz, R., Rocha, M., Rio, M., Sousa, P.: Evolutionary symbiotic feature selection for email spam detection, vol. 1 (2012)

[65] Yeoh, J.M., Caraffini, F., Homapour, E., Santucci, V., Milani, A.: A clustering system for dynamic data streams based on metaheuristic optimisation. (2019)

[66] Aydogdu, O., Ekinci, M.: An approach for streaming data feature extraction based on discrete cosine transform and particle swarm optimization. Symmetry **12**, 299 (2020). https://doi.org/10.3390/sym12020299

[67] Kumar, P., Batra, S.: Meta-heuristic based optimized deep neural network for streaming data prediction. (2018). https://doi.org/10.1109/ICACCCN.2018.8748691

[68] Karimi, Z., Abolhassani, H., Beigy, H.: A new method of mining data streams using harmony search. Journal of Intelligent Information Systems **39**, 491–511 (2012)

[69] Abdulkarim, S.A., Engelbrecht, A.P.: Time series forecasting using neural networks: Are recurrent connections necessary? Neural Processing Letters, 2763–2795 (2019). https://doi.org/10.1007/s11063-019-10061-5

[70] Abidi, M.H., Mohammed, M.K., Alkhalefah, H.: Predictive maintenance planning for industry 4.0 using machine learning for sustainable

manufacturing. Sustainability **14**(6), 3387 (2022)

[71] Izidio, D.M., de Mattos Neto, P.S., Barbosa, L., de Oliveira, J.F., Marinho, M.H.d.N., Rissi, G.F.: Evolutionary hybrid system for energy consumption forecasting for smart meters. Energies **14**(7), 1794 (2021)

[72] Adnan, A., Muhammed, A., Abd Ghani, A.A., Abdullah, A., Hakim, F.: Hyper-heuristic framework for sequential semi-supervised classification based on core clustering. Symmetry **12**(8), 1292 (2020)

Mike Riess

School of Economics and
Business,
Norwegian University of
Life Sciences (NMBU),
P.O Box 5003
N-1432 Ås, Norway

Telephone: +47 6496 5700
Telefax: +47 6496 5701
e-mail*: hh@nmbu.no*
http:/www.nmbu.no/hh

Thesis number: 2022:21
ISSN: 1894-6402
ISBN: 978-82-575-1896-7

Mike Riess was born in Odense, Denmark. He holds a BSc. in Business administration and an MSc. in Business intelligence, both obtained at the School of Business and Social Sciences at Aarhus University in Denmark in 2015 and 2017, respectively.

This Ph.D. thesis addresses problems related to proactive methods of decision support in business processes. These methods include predictive process monitoring, which aims to warn about potential problems before they occur, and prescriptive process monitoring, which seek to proactively remedy predicted issues before they materialize. Four different studies are performed with the aim of improving methods within this area.

Paper one addresses the issue of early warning performance in predictive process monitoring. Three temporally weighted loss functions are proposed with the aim of improving *earliness performance* of remaining time predictions. The results indicate that temporal weighting can indeed improve the performance of early predictions from event-log data.

Paper two offers an alternative to the traditional approach to model evaluation commonly used in predictive process monitoring, by proposing an open-source simulation framework for the generation of synthetic event-log data.

Paper three addresses the issue of customer loyalty in customer service settings. A prescriptive approach to queue management is proposed with the aim of improving customer loyalty. Through an agent-based Monte Carlo simulation model calibrated from the data of a case company, results suggest that the method can improve customer loyalty in situations with insufficient capacity.

Paper four addresses the issue of concept drift, which refers to the long-term performance of machine learning models. Through a literature review, an overview of metaheuristic optimization methods for the automated adaption of Machine learning models is provided, which can be used to guide future research in this area.

Main supervisor: Prof. Joachim Scholderer

Mike Riess currently works as a Senior Data scientist in the Business intelligence and Analytics team at Telia company.

ISBN: 978-82-575-1896-7
ISSN: 1894-6402

Norwegian University
of Life Sciences

Postboks 5003
NO-1432 Ås, Norway
+47 67 23 00 00
www.nmbu.no

116774 / ANDVORDGRAFISK.NO