Improving On-Time In Full Delivery Performance through the Use of Predictive and Prescriptive Process Monitoring Methods

van Wijlen, M.V.

Award date:
2022

Eindhoven University of Technology

Department of Industrial Engineering and Innovation Sciences
Information Systems Research Group

Improving On-Time In Full Delivery Performance through the Use of Predictive and Prescriptive Process Monitoring Methods

Master Thesis

Manou Veronique van Wijlen

TU/e supervisor:
Dr. Laura Genga
Prof.dr.ir. Remco Dijkman

Bright Cape supervisor:
Mart Althuizen

In partial fulfilment of the requirement of the degree of Master of Science in Operations Management and Logistics

Eindhoven, November 15, 2022

Abstract

This master thesis aims at improving on-time in full delivery performance for items within the purchase-to-pay process. A client of data consultancy company Bright Cape is used as use case within this research, to be referred to as Company X. Hence, this research consists of two parts: (1) the predictive process monitoring part, focused on developing and evaluating deep learning models to predict the delivery date for purchased items, and (2) the prescriptive process monitoring part, concerning the creation of a prescriptive model prioritizing purchased items from suppliers that need to be expedited for the day. Five different neural network architectures were tested for predicting the number of days an item would be delivered later or earlier than the contractual delivery date. By adding this prediction output to the contractual delivery date, the predicted delivery date could be received. The CNN seemed to perform best when considering the prediction accuracy versus computational efficiency trade-off. Furthermore, a prescriptive model was developed based on the existing expediting framework of Company X. It can be concluded that the combination of the CNN and prescriptive model allows Company X to act proactively regarding the delivery of purchased items and make positive impact on the on-time in full delivery performance.

Executive Summary

This master thesis is conducted at Bright Cape, a data consultancy company. Bright Cape has selected one of their clients as use case for this research. The selected client is a global mining company, referred to as Company X. Within Company X it is found that the on-time in full (OTIF) delivery rate is insufficient for purchased items from external suppliers. Since 40% of the purchased items is needed for maintenance operations, the insufficient OTIF rate has a negative impact on the maintenance completion of Company X. In particular, late delivered critical materials can disrupt their maintenance processes in the warehouses. To mitigate the effects of late deliveries, Company X calls the suppliers regarding the purchased items that are overdue. This process is called expediting. In order to improve the OTIF rate for purchased items, Company X wants to be able to act more proactively and intervene before the purchased items are actually delivered too late. Moreover, they want to increase the impact of expediting on the OTIF rate by improving the decision making on which items to prioritize for expediting. Therefore, the main objective of this thesis is to provide Company X with tools and recommendations that empowers their expediting team to act proactively regarding the delivery of purchased items and improve expediting prioritization, which can in turn increase the OTIF rate. Hence, this research consists of two parts: (1) the predictive process monitoring part, focused on developing and evaluating deep learning models to predict the delivery date for purchased items, and (2) the prescriptive process monitoring part, concerning the creation of a prescriptive model prioritizing purchased items from suppliers that need to be expedited for the day.

First, a business understanding phase was performed to investigate the purchase-to-pay (P2P) process and expedite process, as well as to determine the predictive and prescriptive task. Subsequently, a data understanding phase was executed to understand the available data and to determine possible input features for the prediction and prescriptive task. The results are shown in Table 1.

Table 1: Results business & data understanding phase

Concept	Result
P2P process	Consists of these main steps: (1) create purchase requisition, (2) release purchase requisition, (3) create purchase order item, (4) release purchase order, (5) receive order confirmation, (6) vendor creates invoice, (7) receive advance shipment notice, (8) record goods receipt, (9) record invoice receipt, and (10) clear invoice
Prediction task	Predict the expected delivery date of purchased items. Model target: number of days between the contractual delivery date and goods receipt event (actual delivery date). The expected delivery date could then be obtained by adding this number of days to the contractual delivery date.
Selected input features predictive models	The P2P process activities, time since case start, time since last event, time since midnight, weekday, vendor, plant, BU, item, material/part number, OTIF of last year, PO lines per vendor, number of changes done to the delivery date, and number of days between PO creation and the contractual delivery date
Expedite process	Consists of determining the priority list for items to expedite. This is done by calculating an expedite score for all overdue items within the expedite dashboard of Compnay X in Celonis. The expedite score is calculated using a weighted scoring model, taking into account the factors: (1) material criticality, (2) stock versus non-stock, (3) supplier historical delivery performance, and (4) days overdue.
Prescriptive task	Generating a priority for expediting (expedite score), for each of the PO items, based upon the predicted delivery date and other relevant features.
Selected input features prescriptive model	Predicted delivery date, material criticality, stock versus non-stock, days overdue score, OTIF of last month, and OTIF of last year

Thereafter, the predictive process monitoring part of the thesis was performed.

Predictive Process Monitoring

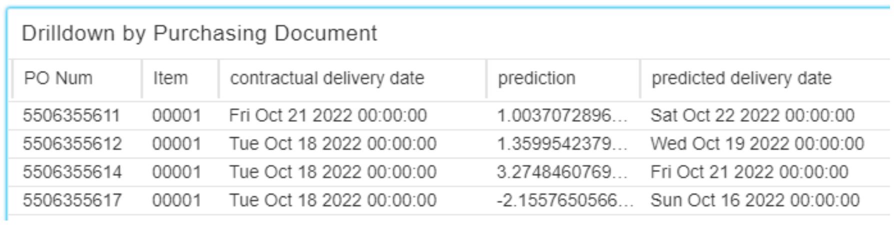
This phase resulted in deploying a CNN model to predict the expected delivery date for items in the P2P process. To achieve this, the following phases were followed based upon the the Business Process Monitoring and Prediction Procedure Model (BPMP-PM) from Becker et al. (2020).

A) *Data Preparation.* In order for the predictive models to understand the input data and use it effectively, the input data set had to be further preprocessed. First, the categorical features were encoded using one-hot encoding or embedding. Secondly, the numerical features were log-scaled to reduce skewness. Next, subsamples were created from the total dataset to be able to conduct experiments within reasonable time. Subsequently, the data in all subsamples and the total dataset was transformed into a prefix format to capture the sequential structure of activities in the log. To evaluate the models, the prefix data was split into a train, validation and test partition. Lastly, random over- and undersampling techniques were used to mitigate the effect of extreme and rare target values on predictive performance.

B) *Model Selection & Implementation.* Five neural network (NN) approaches were selected and subsequently implemented for the defined prediction task and were based on promising results in literature: the MLP (Venugopal et al., 2021), GCN (Venugopal et al., 2021), LSTM (Tax et al., 2017), CNN (Pasquadibisceglie et al., 2020), and BIG-DGCNN (Chiorrini et al., 2021). Besides, a Random Forest approach was added to these selected NN approaches. This approach investigated whether first classifying the extent to which a PO item would be delivered late/early with a RF followed by predicting the exact delivery date with the best performing NN would mitigate the negative effects on predictive performance due to extreme values for the target variable.

C) *Evaluation.* It could be concluded that the CNN was the best predictive model to implement for Company X when making a trade-off between prediction accuracy and computational efficiency. The CNN has a mean absolute prediction error (MAE) of 7.619 days for the prefixes in the test partition of the total dataset. This means that the predicted delivery date is on average 7.619 days earlier or later than the actual delivery date. The goal of Company X was to have a MAE of between 2 and 4 days. It was found that for 50% of all prefixes in the test partition of the total datset, the absolute prediction error was less than or equal to 3.673 days. This was confirmed to be sufficient by the supply chain manager of Company X.

D) *Deployment.* This CNN was subsequently deployed in the machine learning workbench of Company X by writing a Python script transforming the input data for new cases into the right format and predicting the number of days between contractual delivery date and actual delivery date for the purchased items with the trained CNN. The result is shown in Figure 1.


Drilldown by Purchasing Document

PO Num	Item	contractual delivery date	prediction	predicted delivery date
5506355611	00001	Fri Oct 21 2022 00:00:00	1.0037072896...	Sat Oct 22 2022 00:00:00
5506355612	00001	Tue Oct 18 2022 00:00:00	1.3599542379...	Wed Oct 19 2022 00:00:00
5506355614	00001	Tue Oct 18 2022 00:00:00	3.2748460769...	Fri Oct 21 2022 00:00:00
5506355617	00001	Tue Oct 18 2022 00:00:00	-2.1557650566...	Sun Oct 16 2022 00:00:00

Figure 1: The predicted delivery dates for purchased items within the Celonis

It is recommended to keep monitoring the prediction error while the CNN is implemented, in order to see whether the prediction accuracy starts decreasing after a certain period. This could be an indicator of concept drift. This is when the relation between the feature and target variables changes due to external factors, which causes a decrease of prediction accuracy over time. Hence, Park and Song (2020) stress the importance of adapting the implemented predictive model in an online manner if prediction accuracy becomes too low. This can be done by retraining the CNN on more recent data.

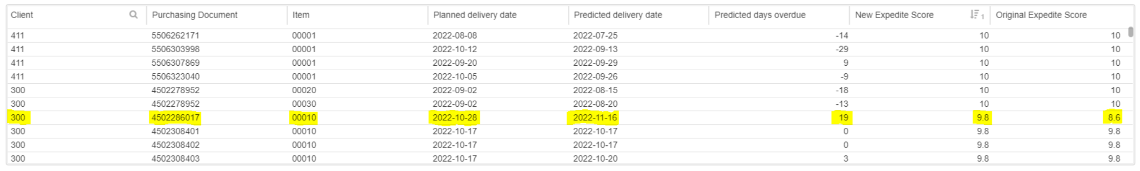
After the predictions for the delivery date could be generated with the CNN, the prescriptive process monitoring part was continued.

Prescriptive Process Monitoring

This phase resulted in the deployment of a prescriptive model that recommends which purchased items from suppliers need to be prioritized for expediting for the day, taking into account the predicted delivery date for purchased items. In the following, the most important steps that had to be performed are explained, again based upon the BPMP-PM Becker et al. (2020).

A) *Method Selection & Implementation.* The newly developed prescriptive model was based upon the existing expedite framework of Company X which consists of an expedite dashboard, showing the prescriptive input data as well as generated priorities for the purchased items, and weighed scoring model to calculate an expediting priority score. Several changes were made to the framework, which finally resulted in the newly developed prescriptive model. First of all, the range of items being expedited was increased by adding an extra filter to the dashboard such that not only the already overdue items can be selected, but also the items that will become overdue within two weeks and are predicted to be delivered more than 1 day late. Subsequently, for all these items a new expedite score is calculated where the factor 'days overdue' was changed such that it took the predicted delivery date into account as produced by the CNN. This resulted in new values for the expedite score for the purchased items which leads to a new and different priority list for expediting.

B) *Deployment.* The prescriptive model was evaluated by means of a past and future analysis. However, the future analysis required the model to be deployed. Therefore, this step is explained before the evaluation step. The prescriptive model was deployed within the Celonis environment of Company X. Since the predictive model was deployed within Celonis as well, the values for the predicted delivery date for items could be retrieved within the existing expedite dashboard. Next, the filter 'orderline overdue' was changed in the expedite dashboard to be able to add the not yet overdue items to the expedite priority list. Moreover, a new formula for the expedite score was added within the expedite dashboard including the updated computation for factor 'days overdue'. An example of the newly generated priority list for expediting after deploying the prescriptive model is shown in 2. It can be noticed that one of the most urgent items for expediting is now a purchased item that is not yet overdue but predicted to be delivered very late.



Client	Purchasing Document	Item	Planned delivery date	Predicted delivery date	Predicted days overdue	New Expedite Score	Original Expedite Score
411	5506262171	00001	2022-08-08	2022-07-25	-14	10	10
411	5506303998	00001	2022-10-12	2022-09-13	-29	10	10
411	5506307869	00001	2022-09-20	2022-09-29	9	10	10
411	5506323040	00001	2022-10-05	2022-09-26	-9	10	10
300	4502278952	00020	2022-09-02	2022-08-15	-18	10	10
300	4502278952	00030	2022-09-02	2022-08-20	-13	10	10
300	4502286017	00010	2022-10-28	2022-11-16	19	9.8	8.6
300	4502308401	00010	2022-10-17	2022-10-17	0	9.8	9.8
300	4502308402	00010	2022-10-17	2022-10-17	0	9.8	9.8
300	4502308403	00010	2022-10-17	2022-10-20	3	9.8	9.8

Figure 2: Expediting priority list based on new expedite score

C) *Evaluation.* The performance of the prescriptive model was evaluated by means of a historical and future analysis. The past analysis showed that for around 77% of the past expedited items it could have been predicted ahead of time that the items would arrive late. If the new prescriptive model was implemented at that time, these items could have been expedited before they would already be overdue. Furthermore, the future analysis showed that around 10% of the not yet delivered most critical purchased items would become overdue soon and were predicted to be delivered late. With implementing the new prescriptive model, these items are added to the expedite priority list, as well as assigned a higher priority for expediting in case the predicted delivery date is later than the planned delivery date. This could prevent late deliveries from happening or from being very extreme by contacting suppliers earlier. This reduces risk since those items are very critical to be able to perform the maintenance activities and are of high order value. A critical note to the prescriptive model is that it does not take into account the accuracy of the CNN. Therefore, items could be incorrectly included into the expediting priority list as they were incorrectly predicted to be delivered too late. Moreover, it affects the reliability of the new expedite score and can lead to a bias within the generate expediting priority list. Therefore, it is recommended to monitor the effect of the prediction error on the expedite score while validating the deployed model, to get an understanding of whether items are wrongly prioritized. This could for example indicate whether the new expedite score needs to be changed.

To summarize, the results show that by implementing the CNN as well as the new prescriptive model using the generated predictions by the CNN, Company X is enabled to act proactively and intervene before items are delivered late. The CNN creates insight into which items are expected to be delivered late. Hence, by using the new prescriptive model these items can be included within the priority list for expediting and a higher priority can be assigned to items which are predicted to arrive later than the planned delivery date. Therefore, Company X can call suppliers before the items are actually overdue. Moreover, by giving more priority to items for expediting which are not overdue yet but are predicted to arrive very late, impact can be made on the OTIF rate since those items can be prevented from actually arriving late or from arriving extremely late.

Preface

I would like to dedicate this section to the people who supported me during this final part of my master.

First of all I would like to thank everyone from Bright Cape who supported and helped me during this project. I would like to thank Ad for supervising me the first months of my thesis. You have been really helpful by learning me how to approach the management of such a big project and to coordinate the visions of Company X, Bright Cape, and the TU/e. Moreover, you have been a great sponsor for this project by letting everyone at Bright Cape know how important it is to develop and do research in the area of predictive and prescriptive process monitoring. Furthermore, I would like to thank Pim for bringing in Company X as a use case, for helping to get the data available, and the support regarding project management. Moreover, I would like to thank Femke for being my technical sparring partner within Bright Cape. Besides, for helping me with the implementation of the predictive model within the machine learning workbench within Celonis, as well as with the creation of the prescriptive model. On top of that, I would like to thank Mart for stepping in as my company supervisor after the leave of Ad. It was great having you to discuss technical details and to help me prioritize tasks within this project.

Next, I want to thank my supervisors from the TU/e. I would like to thank Laura for not only being my supervisor for this project, but also for being my mentor throughout the two full years of my master. Thank you for supporting me in the choices I made regarding my master courses and for teaching me more about the field of process mining. Furthermore, it was great having you as my supervisor for my thesis. Your research interests match really well with my own interests, and I am very excited about the topic for this master thesis. Thank you for meeting regularly during this project, and for always giving me interesting and useful directions to pursue at points where I was stuck. Moreover, I would like to thank you for connecting me with Mozhgan and Andrea. Hence, I would like to thank Mozhgan for enabling me to run my models on the server and also for explaining how to run Python models on the server. Furthermore, I would like to thank Andrea for taking the time to explain his implementation of the BIG-DGCNN.

Contents

List of Abbreviations

AM Alpha Miner. 27, 29

ANOVA Analysis of Variance. 28, 33, 35

BIG Building Instance Graph. 50, 71, 112

BIG-DGCNN Building Instance Graph - Deep Graph Convolutional Neural Network. iii, vi, 47, 49, 50, 61, 65, 66, 67, 68, 69, 70, 71, 72, 83, 84, 86, 88, 107, 111, 113, 119

BPMP-PM Business Process Monitoring and Prediction Procedure Model. ii, iv, 6, 105

BU Business Unit. ii, 24, 32, 33, 35, 36, 74, 83, 107, 108, 109, 110, 111, 112, 115

CDF Cumulative Distribution Function. 59, 60, 86, 105

CI Confidence Interval. 59, 68

CNN Convolutional Neural Network. i, ii, iii, iv, v, 12, 13, 15, 47, 49, 61, 65, 66, 67, 68, 69, 70, 71, 72, 74, 75, 78, 79, 80, 82, 83, 84, 85, 86, 87, 107, 111, 113, 119, 123

CRISP-DM Cross-Industry Standard Process for Data Mining. 6

DL Deep Learning. 2, 3, 4, 5, 6, 9, 12, 13, 14, 15, 27, 40, 43, 83, 88

DNN Deep Neural Network. 9, 15

EDA Exploratory Data Analysis. 7, 27, 39

FN False Negative. 54, 55

FP False Positive. 54, 55

FTE Full Time Equivalent. 1, 125

GAN General Adversarial Network. 88

GCN Graph Convolutional Neural Network. iii, 12, 13, 15, 47, 48, 49, 61, 63, 64, 65, 67, 72, 83, 84, 108, 117, 123

GNN Graph Neural Network. 15

GRU Gated Recurrent Unit. 15

HM Heuristics Miner. 27, 29, 96

IM Inductive Miner. 27, 29

IMf Inductive Miner infrequent. 27, 29, 84, 96

KPI Key Performance Indicator. 1, 5, 9, 16, 17, 19, 80, 124

LRCN Long-term Recurrent Convolution Network. 15

LSTM Long Short-Term Memory. iii, 12, 13, 15, 47, 48, 49, 61, 64, 65, 66, 67, 72, 83, 84, 88, 109, 118, 119, 123

MAE Mean Absolute Error. iii, 53, 54, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 74, 82, 84, 86, 108, 109, 110, 111, 112

MAPE Mean Absolute Percentage Error. 54

ML Machine Learning. xi, 2, 5, 9, 16, 42, 51, 56, 71, 74, 75, 88

MLP Multi-Layer Perceptron. iii, 13, 15, 47, 48, 61, 62, 63, 64, 65, 67, 72, 83, 84, 88, 107, 117, 118, 123

MSE Mean Square Error. 11

NN Neural Network. iii, 9, 11, 12, 15, 16, 41, 45, 47, 48, 49, 50, 51, 52, 53, 54, 61, 62, 64, 65, 66, 67, 68, 69, 70, 71, 72, 74, 83, 84, 86, 88, 106, 107, 108, 110, 117, 119, 122, 123

OTIF On Time In Full. ii, v, 1, 2, 5, 6, 19, 20, 23, 24, 30, 31, 33, 34, 35, 36, 37, 70, 77, 78, 81, 83, 85, 86, 87, 107, 108, 109, 110, 111, 112, 115, 124, 125

P2P Purchase-to-Pay. ii, 1, 2, 5, 6, 7, 9, 17, 18, 19, 20, 21, 24, 26, 27, 28, 29, 30, 32, 34, 47, 50, 61, 68, 69, 83, 84, 86, 87, 117, 118, 120, 121

PBPM Predictive Business Process Monitoring. 2, 3, 4, 5, 6, 7, 9, 12, 13, 17, 27, 41, 48, 53, 83, 88

PM Process Mining. 2

PO Purchase Order. ii, iii, 2, 3, 4, 5, 6, 9, 17, 18, 19, 20, 21, 22, 24, 26, 27, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 47, 51, 52, 55, 56, 61, 68, 71, 74, 75, 77, 78, 79, 80, 81, 82, 83, 84, 85, 87, 107, 108, 109, 110, 111, 112, 115, 124

PR Purchase Request. 5, 19, 21, 22, 124

PrBPM Prescriptive Business Process Monitoring. 2, 3, 4, 5, 6, 8, 9, 16, 17, 83, 85

RF Random Forest. iii, 51, 52, 53, 55, 56, 58, 62, 71, 72, 73, 74, 84, 88, 114, 115, 116

RMSE Root Mean Squared Error. 54, 66, 67, 70

RNN Recurrent Neural Network. 12, 13, 15, 48

SMAPE Symmetric Mean Absolute Percentage Error. 54, 67, 70

TN True Negative. 54, 55

TP True Positive. 54, 55

TPE Tree-structured Parzen Estimator. 52

List of Figures

List of Tables

1 Introduction

This chapter introduces the problem context of this master thesis including the problem statement, as well as the research questions that will be answered and the defined research scope. Lastly, an elaboration is provided of the different chapters framing the content of this research.

1.1 Problem Context

Bright Cape is a consultancy & solutions company that improves the processes of their clients using data science. Recently, Bright Cape has observed that several clients have the problem of lacking insight into their supply chain processes, and in particular into the influence of activities happening in one part of the supply chain on another part in the supply chain. This is a relevant issue, because in order to control the end-to-end supply chain, it is important to have full transparency from supplier to the end customer. Bright Cape has selected one of these clients as use case for this research. The selected client is a global mining company, referred to as "Company X", with as key activities to market a diverse range of metals and minerals (e.g. iron ore and copper) to their customers by finding, mining and extracting, processing, and transporting these substances. Due to confidentiality reasons, this selected client has been anonymized. The supply chain department of Company X ensures the availability of the appropriate raw materials (e.g. explosives and drills), spare parts, and supporting goods throughout the supply chain. According to the supply chain manager at Company X, their supply chain consists of only a few customers buying large quantities of their product, and a huge number of suppliers delivering ordered quantities of purchased items. Company X experiences in particular difficulties regarding the purchase-to-pay (P2P) process within the supply chain. This is the process of moving goods from supplier to the plants. Hence, this research focuses on the P2P process of Company X.

An important bottleneck in the P2P process of Company X is that a high percentage of purchased items from suppliers is not delivered on-time, as stated by their supply chain manager. Since 40% of the purchased items is needed for maintenance operations, late deliveries have a negative impact on the maintenance completion of Company X. In particular, late delivered critical materials can disrupt their maintenance processes in the warehouses. In order to mitigate the effects of those late deliveries, a supplier is being called when the purchased items from that supplier are not delivered on time. This process is called expediting. Within the supply chain department, a team of approximately ten full time employees (FTEs) is dedicated to this expediting process and looking at all orders that are overdue. However, there are two main problems regarding this expediting process. First of all, expediting is only performed for items that are already overdue. But, Company X wants to be able to act proactively and intervene before the order is actually delivered too late in order to greatly improve the rate of items being delivered on-time. Secondly, only a limited capacity is available for the number of suppliers that can be called each day. Therefore, it is important to make the right decisions on which suppliers should be prioritized for expediting in order to make the biggest impact on the on-time in full (OTIF) delivery performance for purchased items.

Accordingly, the goal of this research is to increase transparency in the P2P process of Company X and improve decision-making with respect to expediting, in order to improve OTIF delivery performance for purchased items. Hence, this requires (1) to have a clear overview of the P2P process, (2) to be able to monitor the items throughout this process, as well as (3) to have an estimation of what is going to happen with ongoing orders for purchased items within the P2P process. This enables Company X to act on-time in case something seems to go wrong with an order *or* suboptimal with respect to a specific key performance indicator (KPI), like the on-time delivery rate. Moreover, to support real-time decision-making regarding expediting, it should be possible to not only make accurate predictions for orders of purchased items, but to also know how to use these predictions to deliver recommendations for expediting. This, in turn, should facilitate risk management, reducing unnecessary cost and inefficiencies going along with the business operations.

The area of research which is focused on gaining more process transparency and enhancing process-related decisions is process mining (PM). The goal of process mining is to discover, monitor, and improve actual processes by deriving knowledge from event logs available through companies' information systems (Van der Aalst, 2016). These event logs show records of the execution of a business process (Kirchmer et al., 2017). Process mining is then a good solution to the need of increasing transparency in Company X's P2P process, since it can automatically construct a business process model that captures the P2P process. This visualization helps to represent the flow of activities within the process, such that the process can be analyzed, monitored, and improved more effectively and efficiently. Nowadays, process mining is able to go beyond process exploration and monitoring by applying predictive analytics to process monitoring. This area within process mining is called Predictive Business Process Monitoring (PBPM), and focuses on forecasting the future progress of an ongoing case within the process (Maggi et al., 2014). PBPM has previously been used to make predictions for cases like what will be the next activity or set of activities (Pasquadibisceglie et al., 2019), when will the next event happen (Tax et al., 2017), how much time is needed to finish the case (Park and Song, 2019), or what is the most likely outcome class (Wang et al., 2019). A recent trend in literature is the use of PBPM approaches based on deep learning (DL) architectures, due to the highly promising performance shown in multiple studies, where they often outperformed traditional model-based as well as ML approaches (Tax et al., 2018; Chiorrini et al., 2022). This shows the potential of DL-based approaches for predicting the remaining running time of the ongoing orders for purchased items to identify those that will be delivered late. Therefore, in this thesis we propose to use this technique for on-time delivery prediction, in particular predicting the delivery date for purchased items, to help Company X improve delivery performance regarding their P2P process. Furthermore, Weinzierl et al. (2020a) stress that in order to make the predictions of PBPM techniques more beneficial for process stakeholders Prescriptive Business Process Monitoring (PrBPM) approaches can be used. These approaches assess predictions regarding their impact on the process performance to determine if and when to trigger an action or intervention at runtime, to maximize performance or prevent negative outcomes (Weinzierl et al., 2020a; Shoush and Dumas, 2021). Until now, most research has focused on process monitoring and predictive analytics, and the potential of applying PrBPM techniques was overlooked (de Leoni et al., 2020). Hence, in this thesis we propose to use this technique to assess the predictions for the delivery date of purchased items regarding their impact on OTIF performance to determine which items to prioritize for expediting.

1.2 Research Questions

The goal of this research is to increase transparency in the P2P process of Company X and improve decision-making with respect to expediting, in order to improve OTIF delivery performance for purchased items. This need is derived from observations of Company X, where it is found that the OTIF rate for purchased items is insufficient. This leads to the following main research question:

How can the expected delivery date for purchased items be predicted and actions be recommended regarding expediting in order to increase the OTIF rate?

The main research question addresses the two main problems of Company X as explained in the previous section. Firstly, Company X needs to be able to act proactively and therefore a predictive part is included focused on finding the expected delivery date. Secondly, in order to improve the decision-making at Company X, a prescriptive part is added that aims at finding which actions to recommend regarding expediting. The predictive part needs to be developed first, where subsequently the prescriptive part can be build further upon. The approach that will be considered for the predictive part is based upon PBPM techniques and the approach for the prescriptive part is based upon PrBPM techniques. By applying PBPM techniques the expected delivery date can be predicted for ongoing purchase order (PO) items by using the information of events that have happened for the item up until a certain point in time in the P2P process. The ordering of events will be taken into account when performing the prediction, which is expected to improve the predictions for the delivery date. Then, like explained in the problem context (Section 1.1), PrBPM techniques focus on transforming the generated predictions by PBPM techniques into actionable insights, and are therefore selected to continue upon the predictive part.

In order to answer this question, multiple sub-questions need to be answered.

Before building the actual predictive model(s), an event log needs to be created, and relevant case and event attributes need to be chosen for the prediction task. Therefore, answering of the research question starts with finding out which features to use for the predictive model(s). This leads to the first sub-question:

1. Which data attributes are available, and how can additional features be engineered from these attributes, that might be influential on delivery performance and can be used as input for the predictive algorithm(s) to predict the expected delivery date?

Goal: The goal of this sub-question is to investigate and understand which attributes present in the data might influence the delivery date of a PO item as well as the features that can be generated from these attributes.
Deliverable: An event log including the chosen relevant attributes and engineered features.

Next, the method to be used for the predictive model(s) needs to be selected. Like explained in the problem context (Section 1.1), DL methods are most promising in terms of predictive performance in the field of PBPM. This research aims at investigating which of these promising DL methods is best for predicting the delivery date for purchased items. Therefore, multiple DL models will be selected to be used in the experiments. This sets about the subsequent sub-question:

2. Which DL-based PBPM techniques are most appropriate to predict the expected delivery date for purchased items by Company X?

Goal: This sub-question aims at selecting multiple predictive algorithms from the set of DL-based PBPM techniques explored in the literature study by van Wijlen (2022) that are suitable for the prediction task as well as the process and event log at hand.
Deliverable: Predictive algorithms being selected from the methods explored in the literature review (van Wijlen, 2022). The selection will be based upon mined process models, process characteristics, available data, and the prediction task.

After a choice has been made for which predictive algorithms to use, the data needs to be pre-processed and the features need to be encoded in order for the algorithm to understand and use the data effectively. This gives rise to the next sub-question:

3. Which pre-processing steps need to be executed and which encoding techniques are most appropriate for the input data and chosen predictive algorithms, and how can these pre-processing and encoding steps be performed effectively?

Goal: The objective related to this sub-question is to understand the required pre-processing steps for the data in the event log as well as to select appropriate encoding techniques for the features in the event log. Moreover, to understand how pre-processing and encoding need to be performed.

When it is clear which pre-processing and encoding steps need to be performed and how, these steps will be executed. This results in the following deliverable:

Deliverable: A pre-processed event log and encoded features ready to be used as input to the predictive algorithms.

Then, when the data is prepared, the actual predictive algorithms can be built, validated, and tested. The following sub-questions deal with these final stages of the predictive part of the research:

4. *How do the selected predictive algorithms need to be redesigned and built to be able to predict the expected delivery date for PO items from the event log and chosen input features?*

Goal: This sub-question focuses on building the predictive algorithms and its goal is to investigate how the selected predictive algorithms needs to be adjusted/built to make it work for our event log and features as well as for the prediction task of generating the expected delivery date for PO items.
Deliverable: The selected DL algorithms are built in Python such that it takes the pre-processed event log and encoded features as input, and subsequently predicts the expected delivery date for each PO item.

5. *How well do the built predictive algorithms predict the expected delivery date for PO items in terms of performance measures?*

Goal: This final sub-question regarding the predictive part of the research focuses on validating and testing the built predictive algorithms. It aims at evaluating the developed predictive algorithms by the use of relevant performance measures.
Deliverable: Calculated values for performance measures, e.g. Mean Absolute Error, on the training, validation, and test dataset.

Thereafter, the prescriptive part will be developed. In order to make impactful decisions based upon the delivery date predictions, a PrBPM technique will be used on top of the best performing DL-based PBPM technique. In the end, the prescriptive algorithm needs to generate a prioritized list of purchased items from suppliers that need to be expedited for the day.

Before actually building the prescriptive algorithm, it is important to understand how the expediting process is performed currently at Company X. This should as well give insight into which data attributes, other than the predicted delivery date, influence whether a certain PO item from a supplier needs to be expedited or not. Subsequently, the relevant attributes have to be chosen for the prescriptive task. Moreover, the data might be enriched with additional features, either directly from existing attributes or by applying some transformations to them. These issues are taken into account in the following sub-question:

6. *Which data attributes are available, and how can additional features be engineered from these attributes, that might be influential on whether a PO item from a supplier needs to be expedited or not?*

Goal: The goal of this sub-question is to investigate and understand which available attributes, as well as the features that can be generated from these attributes, in the data might influence the need for expediting a PO item from a supplier.
Deliverable: A dataset including all relevant features for the prescriptive task.

This data might need to be pre-processed before it can be used within the prescriptive algorithm. Subsequently, the prescriptive algorithm can be designed. The next sub-question deals with this:

7. *What PrBPM technique is able to generate recommendations for which PO items from suppliers to expedite and how are these recommendations achieved?*

Goal: This sub-question aims at designing an appropriate PrBPM technique based upon findings of the literature review by van Wijlen (2022) and discussions with domain experts of Company X. Furthermore, there will be investigated how the prescriptive algorithm can generate recommendations for which PO items from suppliers to expedite.
Deliverable: A designed and developed prescriptive algorithm that takes the predicted delivery dates and other selected features as input, and generates a prioritized list of PO items from suppliers that need to be expedited.

Finally, the developed and designed PrBPM technique should be evaluated. The last sub-question is related to this:

8. How well is the PrBPM technique able to generate recommendations for which PO items from suppliers to expedite in terms of performance measures?

Goal: This last sub-question related to the prescriptive part of the research aims at evaluating the developed prescriptive algorithm by the use of relevant performance measures.
Deliverable: A judgement on the performance of the prescriptive algorithm substantiated by calculated values for performance measures.

In the end, the combination of these questions will lead to answering the main research question.

1.3 Research Scope

As mentioned before, the research aim is to increase transparency in the P2P process of Company X and improve decision-making with respect to expediting. Hence, this research only includes the supply chain process, where the main focus is on the P2P process. Inventory management within the supply chain is included to some extent as well, as expediting decisions partly rely on the available and required stock of material. The decision-making should be supported by forecasting and action recommendation techniques coming from the areas of predictive and prescriptive process monitoring. Note that compared to traditional data mining, it is now essential to take into account the ordering of events within the process executions. This is because this research aims at predicting the delivery date for PO items in the P2P process. The delivery date of a PO item can be regarded as the future outcome of an ongoing PO item, dependent upon previous events that have happened for the item within the P2P process. Predictive process monitoring techniques aim at predicting the future progress of a running case within a process by using the event log as input. This event log consist of records of process executions, and thus ordered events. Subsequently, prescriptive process monitoring techniques use the event log and generated predictions to estimate how performance is affected and generate recommendations to support decision-making. Moreover, the scope is partially defined by the available data. As data regarding delivery performance in the P2P process is available from 2019, events happening in the process before that time will not be taken into account. Furthermore, the supply chain manager indicated that internal purchase orders should be excluded from the analysis. Besides, only purchase orders for goods should be included, and thus, for example not the service related orders. Finally, since it is important for Company X to improve the OTIF rate, only delivery performance within the P2P process is included. As mentioned before, the prediction task is to predict the expected delivery date of purchased items. This means no predictions will be executed related to other KPIs in the P2P process, like rework or PR to PO cycle time.

1.4 Research Contribution

When exploring previous research on the topic of on-time delivery prediction, multiple PBPM approaches are found addressing similar problems. For example, Khan et al. (2019) investigate on-time delivery prediction within the order-to-cash process where different machine learning (ML) techniques were applied to predict whether a customer order would be delivered on-time or be delayed. Furthermore, process mining platform Celonis shows a demonstration of using ML techniques like logistic regression, random forest, and XGBoost, to predict whether a customer order will be delivered too late or not (Celonis, 2019). Moreover, Metzger et al. (2019) apply DL-based approaches of PBPM to predict delays in delivery time for a freight transport process. Although, these examples address similar problems, there are three main properties distinguishing our on-time delivery prediction use case. Herewith, our research contributes to the existing literature in this area.

First of all, this research focuses on applying DL-based approaches of PBPM for on-time delivery prediction in the P2P process. This P2P process is more complex than it might seem at first. The process includes a lot of features, e.g. vendor, plant, material number, number of PO lines, which are a mixture of numerical and categorical features. Especially, the categorical features increase

complexity due to an extremely high number of unique category values for several of these features. However, many previous researches in the field of PBPM used only basic features as input for the predictive models, like the activity feature or engineered time features based on an event's timestamp (Park and Song, 2019; Jalayer et al., 2020; Weinzierl et al., 2020b; Wang et al., 2019; Tax et al., 2017; Pasquadibisceglie et al., 2019). Hence, this research seeks to explore what other features than such basic features can be used in order to capture the complexity of the P2P process and improve predictive performance. Furthermore, it is investigated how these features need to be encoded for the predictive models to understand the data, and how PBPM approaches need to be adapted to be able to take these features as input.
Secondly, this research aims at investigating a special type of DL-based PBPM approach for on-time delivery prediction, that is a graph neural network. It has the potential to exploit the process structure in order to make predictions about the future of a case by combining the graph representation of the process model with the features in the event log (Rama-Maneiro et al., 2021b). From what is found in literature, the concept of exploiting structural properties of a process execution when generating a prediction is still an underdeveloped topic (Chiorrini et al., 2022). Therefore, this research adds to the existing literature by comparing a graph neural network to other DL-based PBPM approaches.
Thirdly, this research explores how generated process predictions can be converted into actionable insights for the Company X. Weinzierl et al. (2020a) mention that, in order to make the predictions of PBPM techniques more beneficial for process stakeholders, Prescriptive Business Process Monitoring (PrBPM) approaches can be used. PrBPM techniques use event logs to predict how a process instance is going to unfold in the future and how performance is affected. These predictions are then used to determine if and when to trigger an action or intervention at runtime to maximize performance or prevent negative outcomes (Weinzierl et al., 2020a; Shoush and Dumas, 2021). Until now, most research has focused on process monitoring and predictive analytics, and the potential of prespcriptive analytics was overlooked (de Leoni et al., 2020). Hence, this research aims at applying PrBPM techniques to support P2P process-related decisions, by providing recommendations for expediting based upon the predicted delivery date for running cases.

Lastly, the contribution of this research to the business, Company X, is discussed. This research aims to improve the OTIF delivery performance by providing Company X with tools and recommendations that empowers their expediting team to act proactively regarding the delivery of purchased items and improve expediting prioritization. The reason for Company X to improve the of on-time in full (OTIF) delivery performance is that 40% of the purchased items is needed for maintenance operations, meaning a higher OTIF rate is expected to improve maintenance completion. This, in turn, is likely to improve the machine availability, leading to a higher number of productive hours, which in the end is proposed to increase the volume of saleable products. Having the tools to predict the expected delivery date for purchased items and to act effectively upon these predictions would allow Company X to contact suppliers ahead on time. Moreover, better schedules for maintenance can be made, for instance, by scheduling some tasks earlier for which material is expected to be available on-time and delaying maintenance tasks for which material is needed that is expected to be delayed.

1.5 Chapter overview

The structure of this master thesis is guided by the overall research methodology, as depicted in the framework in Figure 3. Note that this methodology is based upon the Business Process Monitoring and Prediction Procedure Model (BPMP-PM) from Becker et al. (2020), which is an adaptation of the well-known Cross-Industry Standard Process for Data Mining (CRISP-DM) (Data Science Process Alliance, 2022). The methodology is divided into two parts: (1) the predictive process monitoring, focused on developing and evaluating DL models to predict the delivery date for PO items, and (2) the prescriptive process monitoring, concerning the creation of a prescriptive model prioritizing PO items from suppliers that need to be expedited for the day. The first three steps in the framework, the literature review as well as business and data understanding, are performed for both the predictive and prescriptive part simultaneously, and are therefore located outside the boundaries of the predictive and prescriptive part in Figure 3.

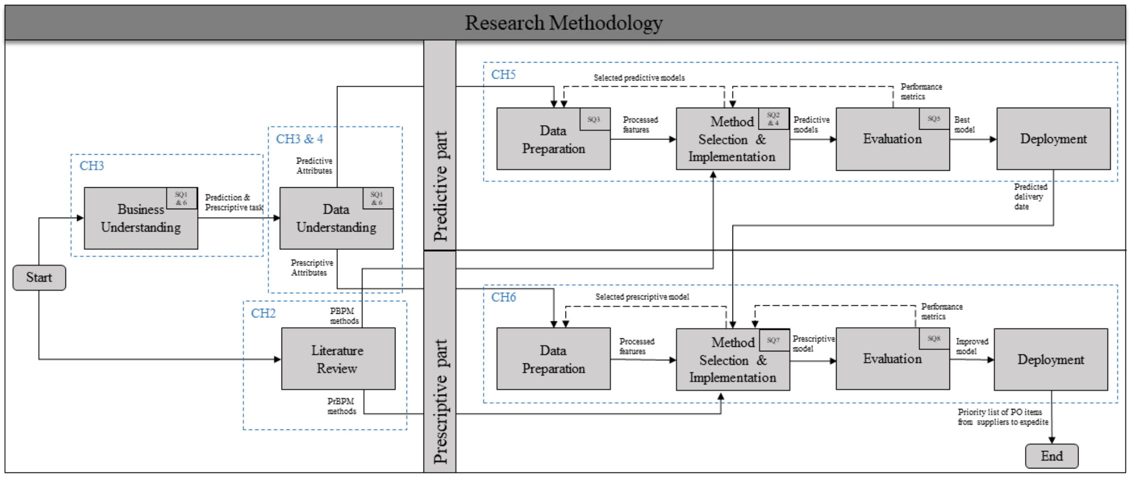


Figure 3: Summary of Research Methodology

Figure 3 also shows the mapping of the chapters and sub research questions (SQ1-8) to the steps in the framework. First, a summary is provided in Chapter 2 of the performed literature review (van Wijlen, 2022) on the state-of-the-art methods in the field of predictive and prescriptive process monitoring. Next, Chapter 3 elaborates on the business understanding phase of the research. Here, the P2P and expedite process are explored and it is determined what to predict and prescribe to realize the business objective. This is achieved by conducting interviews with the supply chain manager of Company X. Moreover, the first steps of the data understanding phase are discussed, which involve the understanding of the available data and its limitations, as well as the investigation of possible input features for the predictive and prescriptive task. Therefore, the data is explored through analyses in Celonis and clarified further by the supply chain manager and process mining expert of Company X. Thereafter, the second part of the data understanding is described within within Chapter 4, where an in-depth analysis is performed of the P2P process characteristics and possible input features for both the predictive and prescriptive models. First, the P2P process characteristics are examined by analyzing multiple Petri Net representations of the process, discovered from the P2P event log. Next, the data for the possible input features is analyzed by means of an exploratory data analysis (EDA). This chapter is followed by Chapter 5, which is dedicated to all the steps in the research framework regarding the predictive process monitoring. First, the performed steps are explained, and subsequently an analysis of the results for these steps is provided. Data preparation includes a) selecting the required data, b) solving the data quality issues as found in the data understanding phase, c) encoding of the categorical and numerical features, d) creating a subsample of the total dataset, e) transforming the cases into a prefix format, f) splitting the data into a train, validation and test set, and g) resampling of the data. Next, the predictive models are selected and built within the method selection & implementation phase, based upon the PBPM methods found in the literature review, chosen predictive task, and selected input data. Subsequently, the best predictive model is selected based on the results for the specified evaluation metrics, and afterwards deployed. Thereafter, Chapter 6 explains the method and results for all the steps of the prescriptive process monitoring part of this research. First, the data is prepared by selecting the required data and possibly solving data quality issues. Next, the prescriptive model is designed and built, based upon the PrBPM methods found in the literature review, the expedite process, the chosen prescriptive task, as well as the predicted delivery date and other selected input data. This model is subsequently evaluated by means of a historical analysis looking at past expedited items, and a future analysis, looking at the impact of the deployed prescriptive model on the expedite process. Finally, the research is completed with a conclusion and discussion in Chapter 7. All the findings for the sub research questions, which were covered in the steps of the research framework, are discussed and lead to an answer for the main research question. Furthermore, the limitations are provided regarding the research, as well as recommendations for Company X, and suggestions for future research.

2 Literature Review

This chapter contains a summary of the performed literature review related to this master thesis (van Wijlen, 2022). The aim of this literature review was to identify and analyze PBPM and PrBPM methods available in literature that can make predictions and based on these predictions generate recommendations or alarms for running process instances to improve process performance and prevent undesired activities. As aforementioned, this master thesis addresses the need for a DL-based approach of PBPM for on-time delivery prediction in the P2P process as well as a PrBPM approach to transform the predictions into a prioritized list of PO items for expediting. Hence, this chapter provides a background on the most important constructs used within the research. These constructs include certain preliminary concepts, deep learning methods for PBPM, and PrBPM methods.

2.1 Preliminary Concepts

This section summarizes the preliminary concepts that are used later in this master thesis. These concepts are PBPM, PrBPM, event logs, and neural networks (NNs).

Predictive Business Process Monitoring (PBPM) is a subfield of process mining which focuses on forecasting what is going to happen with an ongoing case within the process in the future (Maggi et al., 2014). Possible predictions are what will be the next activity or set of activities, when will the next event happen, or how much time is needed to finish the case. According to Tax et al. (2018) many ML techniques have been used to learn a predictive model for the aforesaid purposes, however, approaches based on Deep Learning (DL) are the ones that have obtained the best results. DL is a specific type of ML using Deep Neural Networks (DNNs) as predictive models.

Weinzierl et al. (2020a) mentions that in order to make the predictions of PBPM techniques more beneficial for process stakeholders *Prescriptive Business Process Monitoring (PrBPM)* approaches can be used. These approaches assess predictions regarding their impact on the process performance, usually measured by KPIs, to prevent undesired activities. PrBPM techniques use event logs to predict how a process instance is going to unfold in the future and how performance is affected, and to determine if and when to trigger an action or intervention at runtime to maximize performance or prevent negative outcomes (Weinzierl et al., 2020a; Shoush and Dumas, 2021).

Both PBPM and PrBPM rely on the use of event logs as input. Hence, this concept is explained in the next paragraph. Thereafter, since DL methods for PBPM use neural networks as predictive models, the concept of neural networks is described.

2.1.1 Event Log

The event log that is created from different running processes is used primarily as input for predictive process monitoring techniques (Harane and Rathi, 2020). An example of an event log is given in Table 2.

Table 2: Business event log about medial trials performed in a hospital (Rama-Maneiro et al., 2021a)

Case ID	Activity	Timestamp	Resource
Case2118	Inclusion	14-01-2010 07:52:50	Peter
Case2118	TAC	09-02-2010 13:01:11	Joseph
Case2118	Blood Analysis	17-02-2010 07:44:53	Joseph
Case2118	Intervention	17-02-2010 07:44:59	Joseph
Case2118	Discharge patient	18-02-2010 09:00:10	Joseph
Case2088	Inclusion	04-02-2010 08:37:45	Peter
Case2088	Pneumothorax scan	04-02-2010 09:01:28	Peter
Case2088	TAC	04-02-2010 09:01:35	Peter
Case2088	Blood Analysis	16-03-2010 13:08:40	Peter
Case2088	Discharge patient	31-03-2010 11:08:53	Dio

In the next paragraph the meaning of an event log, traces and sequences is explained formally. Formally, let A, T, E be the set of all possible activities, timestamps and event identifiers. It is assumed for events that they are characterized by various properties, e.g. timestamp, activity and resource. There are functions that assign the attributes to an event, e.g. $\pi_T \in E \to T$ that assigns timestamps to events, and $\pi_A \in E \to A$ that assigns an activity to an event.
An *event log* is a set of events, where each event is linked to one trace and is globally unique. This implies that the same event cannot occur twice in a log. A trace in a log shows the execution of one case (Tax et al., 2017).

The paper by Tax et al. (2017) presents the following definitions for a trace and event log:

Definition 1 (Trace, Event Log) *A trace is a finite non-empty sequence of events $\sigma \in E^*$ such that each event appears only once and time is non-decreasing, i.e. for $1 \leq i < j \leq |\sigma|$: $\sigma(i) \neq \sigma(j)$ and $\pi_T(\sigma(i)) \leq \pi_T(\sigma(j))$. C is the set of all possible traces. An event log is a set of traces $L \subseteq C$ such that each event appears at most once in the entire log.*

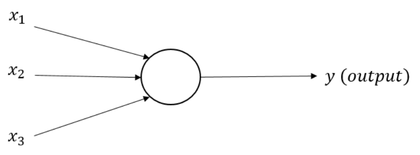
For the sake of simplicity, events in an event log can be represented by just their activity attribute. Within a given activity set A, A^* represents the set of all sequences over A and trace $\sigma =< a_1, a_2, ..., a_n >$ a sequence of length n with $<>$ being the empty sequence and $\sigma_1 \cdot \sigma_2$ the concatenation of sequences σ_1 and σ_2. The prefix of length k $(0 < k < n)$ of sequence σ is $hd^k(\sigma) =< a_1, a_2, ..., a_k >$ and the suffix is $tl^k(\sigma) =< a_{k+1}, ..., a_n >$. For example, for given sequence $\sigma_1 =< a, b, c, d, e >$, $hd^2(\sigma_1) =< a, b >$ and $tl^2(\sigma_1) =< c, d, e >$.

In order to understand the concepts completely, an informal explanation is given as well. The event logs show records of the execution of a business process and can be explained as a series of activities performed by a group of resources to achieve an objective (Kirchmer et al., 2017). Thereby, an event log consists of different cases, where each case represents a particular execution of the recorded process under consideration (Van der Aalst, 2016). The main elements in the event log are the events which are identified by a case identifier, the activity being executed, and a timestamp. Moreover, they can have event attributes, which are specific to each event, or case attributes, which are shared by the events belonging to the same case. The term for a sequence of events from the same case is a trace, and if the sequence of events is still ongoing (the process for a case has not finished yet) it is called a prefix (Aalst et al., 2011).

2.1.2 Neural Networks

Before diving into the details of building a Neural Network, it is important to understand the concept of encoding. In order for any predictive model to understand the input data, the event log should be converted into feature vectors. Simply put, feature vectors are the properties of the events (Harane and Rathi, 2020). These properties are event specific attributes, which are for example

event ID, timestamp, resource and cost. There are different methods for encoding, but in general encoding indicates events and the information related to them (Harane and Rathi, 2020).
A Neural Network (NN) is simply a function mapping inputs to outputs. The concept relates back to the earlier version which is the *perceptron*. It is an element that accepts multiple inputs x_i, $i = 1...N$, and computes a weighted sum of the inputs where for each input the weight (w) can only be -1 or +1. The sum is then compared to a threshold (θ), and an output is produced which is either 1 or 0, depending on whether the weighted sum exceeds the threshold or not (Kanal, 2003). The concept is summarized in Figure 4. Subsequently, multiple perceptrons/sigmoid neurons can be combined to form a layer of perceptrons/sigmoid neurons, and even multiple layers. A neural network consists of one layer of input units, another layer of output units and one or multiple layers in-between which are specified as hidden units (Tax et al., 2017). The first layer is making simple decisions by weighting the inputs. The outputs of this input layer are the inputs for the first hidden layer, where the outputs of the units of each hidden layer form the inputs for the subsequent hidden layer. Next, the outputs of the last hidden layer are the inputs for the output layer (Tax et al., 2017). This is depicted in Figure 5.



$$y = \begin{cases} 0\ if \sum_{i=1}^{N} w_i x_i < \theta \\ 1\ if \sum_{i=1}^{N} w_i x_i \geq \theta \end{cases}$$

Figure 4: Perceptron concept

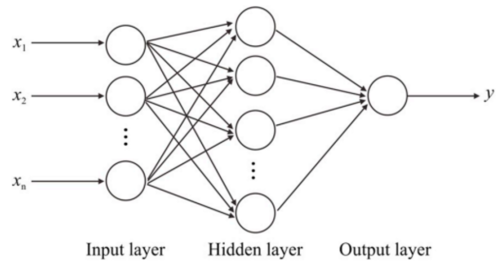


Figure 5: Architecture of a Multi Layer Neural Network (Fath et al., 2020)

The output of each unit (neuron) is a function over the weighted sum of its inputs with a bias added to it. A neuron K can be mathematically defined by the following two equations (Fath et al., 2020):

$$y_k = f(u_k + b_k) \tag{1}$$

$$u_k = \sum_{i=1}^{N} w_{ki} x_i \tag{2}$$

where $x_1, x_2, ..., x_n$ are the inputs and $w_{k1}, w_{k2}, ..., w_{kn}$ are the weights of the neuron, u_k the linear output of the weighted sum, b_k the bias term, f the (non-linear) activation function, and y_k the output of the neuron. The neural network is trained on the basis of the back propagation algorithm. This is a learning procedure based on the error-correction rule. The network produces network outputs by processing the inputs. These outputs are compared against the target values and an error value is calculated. Subsequently, the weights and biases in the model are adjusted to minimize the error value. These values are adjusted based on the gradient of the loss with respect to the weights and biases. The training process finishes when the network reaches a predefined minimum acceptable error. Usually the mean square error (MSE) or cross-entropy is used as error measure (Fath et al., 2020).

2.2 Deep Learning Methods for PBPM

According to Tax et al. (2018) PBPM approaches based on Deep Learning (DL) are the ones that have obtained the best results. Therefore, this literature review focused only on DL methods for PBPM. An overview of the DL methods used for PBPM in the reviewed papers where a real experiment is performed (that are not literature reviews or surveys itself) is given in Table 4. This includes the different neural network architectures, prediction tasks, input features and encoding methods that were used within the reviewed papers.

Neural Network Architectures
Regarding the neural network architectures, it is seen that a main distinction can be made between methods exploiting the sequential nature of the event log data, and methods exploiting the process structure. Besides, there are hybrid methods upcoming combining existing methods to utilize both the process structure and sequential features from the event log to make predictions. The NN architectures being most frequently used for PBPM in literature according to Table 4 are: the Recurrent Neural Network (RNN) with in particular the Long Short-Term Memory (LSTM) architecture, the Convolutional Neural Network (CNN), and the Graph Convolutional Neural Network (GCN). Therefore, these architectures are discussed in more detail in the upcoming paragraphs.

RNN
The Recurrent Neural Network (RNN) is a special type of neural network which is an approach to sequence modeling problems. Due to the *temporal organisation of the activities* in a trace the authors of the reviewed papers seem to prefer recurrent architectures as they are designed to deal with temporally dependent data (Park and Song, 2019; Weinzierl et al., 2020b; Wang et al., 2019; Tax et al., 2017; Hinkka et al., 2018). One of the special RNN architectures is the LSTM, which can model long-term dependencies in a powerful way by including a more complex memory cell in its architecture.

CNN
The reason for the interest in this architecture is according to Pasquadibisceglie et al. (2019) the fact that it has shown very high accuracy in recent literature whenever applied to image data embedding a clear spatial structure. The CNN is composed of model input, feature extraction layers (convolutional and pooling layers), a fully connected layer, and model output (Park and Song, 2020). It is an extension of the most basic feed-forward neural network model by adding three concepts: convolution, pooling, and weight sharing (Pasquadibisceglie et al., 2019). Moreover, the input that is given to the neural network is grid-like, which can either be a 1D grid (Weinzierl et al., 2020b) or 2D grid/image (Pasquadibisceglie et al., 2019; Park and Song, 2020).

GCN
This last NN architecture is especially gaining attention in literature, because it exploits the *process structure* in order to make predictions. The main idea behind the GCN is that it combines the graph representation of the process model with the features for each event of the traces in a graph convolutional layer (Rama-Maneiro et al., 2021b).

It is explained in the literature review of Rama-Maneiro et al. (2021a) that when comparing CNNs against the RNNs for predictive process monitoring, CNNs may have the efficiency advantage of a faster training and inference especially for event logs including longer traces. Yet, RNNs may be better able to capture longer dependencies between the events of the trace as the hidden state for a given event relies on every event before, where with a CNN this depends the s most recent events with s being the size of the kernel. It is then seen in Rama-Maneiro et al. (2021a) that overall regarding neural network type RNNs seem to outperform CNNs. According to the authors this is due to the fact that CNNs also need to set up pooling layers to improve their performance, but this increases the amount of hyperparameters in the model which may impose a suboptimal architecture hindering their performance. Whereas RNNs are just stacked one after another giving good results. However, CNNs perform much faster than RNNs so they may be the only candidate in when used in a case where speed is a priority.

Philipp et al. (2019) state that the GCN approach seems promising as it shows a strong inductive on graph structured data, similar to the effect that can be observed in layers of CNNs on image data. Their results then also show that the GCN is converging faster to a better regression value than a fully connected neural network with only linear instead of graph convolutional layers.
Within the paper of Venugopal et al. (2021) it is shown that their GCN model variants have good performance compared to previous models, like the MLP and LSTM, for predicting the timestamp of the next event. For predicting the next event, the results are varying for different datasets. On the Helpdesk dataset the GCN model variants outperform two out of three LSTM models and the reference CNN model. However, the models perform poorly on the BPI 12 (W) dataset for the event prediction task. The approach of Chiorrini et al. (2022) shows a better prediction accuracy for the next event prediction task with the Helpdesk dataset than Venugopal et al. (2021), however less on the BPI 12 (W) dataset. The authors from Chiorrini et al. (2022) then also conclude that their Deep GCN shows promising performance in datasets capturing a process with a consistent presence of parallelism, like in the Helpdesk dataset, while performing less effectively in sequential datasets, like BPI12 (W).

Prediction Task
As can be seen in the paper overview in Table 4 there are different prediction tasks performed with the DL methods for PBPM. These different prediction tasks have been categorized on two dimensions: prediction level and prediction type. The prediction level is split up into a case/instance-level and a process level. With the predictions made on a case-level a value or class is predicted specifically for one case (e.g. the next activity for the partial trace of the case), whereas with predictions made on a process level a value or class is predicted over the entire process (e.g. mean waiting time of activities in the process). The two prediction types are regression and classification. In the case of regression a value is predicted (e.g. a time-value), and with classification an outcome class is predicted (e.g. whether the outcome of a check in the process for a case will be positive or negative). The observed prediction tasks and their categorization is described in Table 3.

Table 3: Prediction task categorization

	Case/Instance level	**Process level**
Regression	Remaining time Timestamp of next event Output value	Waiting time Processing time Sojourn time
Classification	Next activity Next event Remaining sequence/Full continuation of running case Output class	

It can be concluded that most papers analyzed in this review make predictions on a case/instance-level (Park and Song, 2019; Philipp et al., 2019; Jalayer et al., 2020; Rama-Maneiro et al., 2021b; Venugopal et al., 2021; Weinzierl et al., 2020b; Wang et al., 2019; Tax et al., 2017; Pasquadibisceglie et al., 2019; Hinkka et al., 2018; Weinzierl, 2021; Chiorrini et al., 2022). Only one paper on a process level (Park and Song, 2020). The authors state that it is more practical to predict process performance at the process model level (e.g. predicting the processing or waiting time of activities in the process) in order to detect potential weaknesses in the process and so come up with proactive actions that will improve process performance (Park and Song, 2020). Moreover, it is noted that some papers focus on a single prediction task (Rama-Maneiro et al., 2021b; Weinzierl et al., 2020b; Weinzierl, 2021; Wang et al., 2019; Pasquadibisceglie et al., 2019; Philipp et al., 2019; Chiorrini et al., 2022; Hinkka et al., 2018), and others on multiple prediction tasks (Park and Song, 2019;

Tax et al., 2017; Jalayer et al., 2020; Venugopal et al., 2021; Park and Song, 2020). For example, Tax et al. (2017) leveraged multitask learning by predicting the next activity and its timestamp with the same neural network.

Input features

As displayed in Table 4 different studies use different input features within their DL predictive models. A clear observation is the fact that activity is most used as input, for 9 out of 12 studies this feature is explicitly mentioned (Park and Song, 2019; Jalayer et al., 2020; Philipp et al., 2019; Rama-Maneiro et al., 2021b; Tax et al., 2017; Pasquadibisceglie et al., 2019; Hinkka et al., 2018; Weinzierl, 2021; Chiorrini et al., 2022). Only two of these papers used resource as additional input feature (Weinzierl, 2021; Park and Song, 2019). In 5 papers timestamp was used as input for the prediction task (Rama-Maneiro et al., 2021b; Venugopal et al., 2021; Tax et al., 2017; Pasquadibisceglie et al., 2019; Weinzierl, 2021). Time stamp was not always used directly as a feature but was often encoded into a more useful value, like ”time since previous event”. Some studies just noted ”event attributes/features” as input features (Weinzierl et al., 2020b; Wang et al., 2019; Hinkka et al., 2018). However, as Hinkka et al. (2018) shows, these attributes can differ per event log. Where Hinkka et al. (2018) explains which attributes are included for each event log, Weinzierl et al. (2020b) only shows the number of attributes for each event log and Wang et al. (2019) does not go into detail at all. The attributes ”Organizational resource” and ”lifecycle transition” are found in most of the event logs used by Hinkka et al. (2018). Furthermore, some came up with additional features which were attributes of a beforehand mined process model (e.g. Petri Net). This was especially seen for predictive model architectures requiring graph input data. For example, in Rama-Maneiro et al. (2021b) the marking of a place at given time step is included as input and in Weinzierl (2021) edge attributes as source node, target node and direction are added.

Encoding methods

To be able to use the inputs in a neural network, most inputs need to be modified. The reasoning behind this is the fact that the inner logic of neurons is designed for input vectors with values between -1 and 1 and a mean of 0. The process of transforming the input data to this range is called data encoding. Next, the mostly seen encoding methods in the reviewed literature are discussed.
Firstly, an important distinction between encoding methods is whether the attribute being encoded is categorical or continuous. Encoding is most important for categorical attributes a numerical representation is needed in order to give it as input to the neural network. However, even if the attribute is already numerical as for the continuous attributes, the value is often transformed to a better suiting format, like in Wang et al. (2019) and Weinzierl et al. (2020b) where normalization is applied. According to Rama-Maneiro et al. (2021a) multiple normalization techniques can be used such as min-max normalization, log-normalization, z-score normalization, or tanh-estimators. Each categorical attribute must be encoded in fixed feature vectors individually representing them. The main encoding methods that were used for categorical attributes were one-hot encoding (Park and Song, 2019; Wang et al., 2019; Tax et al., 2017; Hinkka et al., 2018; Weinzierl, 2021; Chiorrini et al., 2022), embedding (Jalayer et al., 2020; Rama-Maneiro et al., 2021b), hash encoding (Weinzierl et al., 2020b) and frequency-based encoding (Pasquadibisceglie et al., 2019). It is seen that one-hot encoding is most popular, followed by embedding.

Table 4: Summary Deep Learning Methods for PBPM

Paper	DL method	Prediction task	Features	Encoding	Architecture
Park and Song (2019)	LSTM-NN	remaining time, next activity	activity, resource	one-hot	2 shared LSTM-layers followed by 2 specialized layers for each prediction task
Philipp et al. (2019)	GCN	output: payment per area (regression)	adjacency matrix, additional event attributes (activity, duration, priority, costs)	graph structure	2 graph convolutional layers (hidden layers) and single input channel, LeakyReLU activation function
Jalayer et al. (2020)	Attention Mechanism	next event/activity (done multiple times for remaining sequence prediction)	activity name	word embedding	encoder-decoder and feed-forward neural network (attention layer)
Rama-Maneiro et al. (2021b)	GNN with RNN (Graph Recurrent Neural Network)	next activity	node feature matrix (marking of a place at given time) & attribute feature matrix (activity fired, encoded bucket of time since previous event, encoded bucket of time since first event of prefix, event attributes: resource)	separately embed the categorical features (length=32)	two GRNN layers, readout phase, stacked LSTM, dense layer with softmax classifier.
Venugopal et al. (2021)	GCN	timestamp of next event, next event	activity ID, time since previous event in case, time since case started, time since midnight, day of the week for the event	Real value taken (no encoding)	GCN layer followed by dropout, two fully connected layers, dropout, and another fully connected layer.
Weinzierl et al. (2020b)	three types of DNN: MLP, LSTM, and CNN (LSTM has hightest predictive quality)	next activity	event attributes	min-max normalization for numerical attributes, 5 techniques for categorical attributes were tried: binary, hash, ordinal, onehot and word embedding (hash leads to highest predictive quality)	MLP: one input layer, four hidden layers (input size = $\{300, 200, 100, 50\}$, ReLU activation function, random dropout of 50%), one output layer (Softmax activation function) & LSTM: input layer, one hidden layer (LSTM layer with output size 100, linear activation, random dropout of 20%), one output layer. & CNN: one input layer, seven hidden layers (1-5: convolutional layer blocks, ReLU activation, 1D max pooling, 6: dense layer of 100 neurons with ReLU activation), and one output layer (Softmax activation)
Wang et al. (2019)	Attention-based Bidirectional LSTM NN (Att-Bi-LSTM)	classification: predict most likely output class	event features	*numerical attribute:* normalization, *categorical attribute*: one-hot	Five layers: input layer, encoding layer, BiLSTM layer, attention layer, output layer
Tax et al. (2017)	LSTM-NN	next activity and its timestamp, full continuation of running case, remaining time	event attributes: activity, 3 time based features (time between previous and current event in trace, time within the day, time within the week)	one-hot encoding for activity, value as-is for time	tested three types: (1) two separate LSTM models (one for next activity, one for timestamp) using same input features, (2) single LSTM model with shared multi-tasks layer generating two outputs, (3) number of shared LSTM layers for both tasks (n) followed by number of specialized layers (m) for each prediction task. Option 2 and 3 perform best.
Pasquadibisceglie et al. (2019)	CNN	next activity	activity and timestamp	2D-Image I_k where k is the length of the prefix and h is the size of the activity domain A of the event log. The rows of the image represent the consecutive indexes of the events in the prefix sorted by timestamp. Columns represent distinctive activities of activity domain. Encoding: Every pixel (x,y) is a 2D-vector with activity count since start of trace and how many days have passed since start of trace until latest occurrence of specific activity.	Input, three pairs of convolutional and max-pooling layers (1: 32 filters of size 2x2, 2: 64 filters with size 4x4, 3: 128 filters with size 8x8, each pooling layer had stride 2 and 2x2 sliding window), fully connected flattening layer, output layer (softmax activation function)
Park and Song (2020)	Three types: CNN, LSTM, combination of LSTM and CNN: LRCN	future performance measures of states and transitions in the business process (waiting time, processing time & sojourn time)	CNN: performance measures at different time windows, LSTM: snapshot of performance on all locations in the business process model, LRCN: performance at a time window	process representation matrix in form of transition system	CNN: two pairs of convolutional layer with pooling layer (filter and pooling size (3,3)), a flattening layer, fully connected output layer. LSTM: two hidden LSTM layers (# cells equals steps in time window), fully connected output layer, LRCM: CNN, LSTM layer, another LSTM layer stacked on it, fully connected output layer.
Hinkka et al. (2018)	GRU (type of RNN)	any case-level prediction task, e.g. next activity or final duration of running case	activity, event attributes, event attribute cluster	one-hot encoding	one-layer GRU with 256 as hidden dimension size.
Weinzierl (2021)	Gated Graph Sequence Neural Network: integrates GRU into GNN	next activity	adjacency matrix, node features (activity, time since previous event in process instance, time since start of process instance, time since midnight, day of the week, resource feature), edge features (source node id, target node id, type of edge)	activity, resource and edge type (categorical attributes) are one-hot encoded	Gated graph layer with four sub-layers (GRU iterations), followed by global attention layer and four dense layers
Chiorrini et al. (2022)	Deep Graph Convolutional Neural Network (Deep GCN)	next activity	activity, flow information from graphs	instance graphs for the traces & one-hot encoding for activity set	input instance graphs (prefixes), subsequently several graph convolutional layers, followed by a SortPoolingLayer, a 1D-convolution layer with drop-out and finally a dense layer and softmax layer.

2.3 PrBPM Methods

It is observed in literature that the developed PrBPM techniques are mainly recommender systems or alarm systems. Moreover, two main goals for the use of a prescriptive technique can be distinguished: optimizing performance (a KPI) or mitigating risk/negative outcomes.

As stated by de Leoni et al. (2020) a Process-Aware *Recommender system* consists of three main elements: (1) monitoring, i.e. keeping track of running process instances, (2) a predictive technique, which forecasts the future outcome (e.g. value, class, or probability) of running instances, and (3) a prescriptive-analytics system providing recommendations on the running instances. Overall, it is hard to generalize the frameworks used for prescriptive recommender systems. As a predictive technique mostly ML methods are being used, however all different ones. Especially, the prescriptive methods are different among the reviewed studies. Examples of models found are a combination of nearest neighbour algorithm and simulation (Weinzierl et al., 2020a), a memory-augmented NN (Khan et al., 2021), and a transition system (de Leoni et al., 2020).

The other type of prescriptive approach is called an *alarm system*. Such a system decides whether an alarm is sent or not to a human worker or software bot that shows which intervention to perform (Fahrenkrog-Petersen et al., 2021). Whether an alarm is raised or not is dependent upon multiple factors. First of all, a predictive model is used to estimate the probability that an ongoing case will end in an undesired outcome. According to Teinemaa et al. (2018) this can be any probabilistic classification algorithm, however, both Teinemaa et al. (2018) and Fahrenkrog-Petersen et al. (2021) experimented with the ML techniques random forest and gradient boosted trees. Next, if this probability is above a certain threshold value an alarm is raised and intervention is triggered. The alarming framework generates alarms such that it minimizes the net cost given an event log, cost model and set of running cases. Therefore, the alarming threshold should be chosen optimally such that it minimizes the net cost on a log with respect to a given probability estimator and alarm model. In both papers this value is found with a hyperparameter optimization technique called Tree-structured Parzen Estimator. The alarm model consists of three elements: (1) a function modeling the cost of intervention, (2) a function modeling the cost of compensation, i.e. cost of doing the intervention while it was not necessary in the end, and (3) a function modeling the mitigation effectiveness of the intervention, i.e. relative benefit of raising an alarm. Then, an alarm *cost* model combines these three elements with a function modeling the cost of the undesired outcome.

3 Business and Data Understanding

This chapter elaborates on the business understanding as well as data understanding phase of this master thesis for both the predictive and prescriptive part. Firstly, the method is explained for the steps that have been performed regarding the business and data understanding. Subsequently, insight is provided into Company X's operations, with in particular a detailed investigation of the P2P process. Subsequently, the available data is discussed as well as the limitations on this front. Moreover, the available data is further investigated to find possible input features for the predictive and prescriptive task. Lastly, the filters are discussed which have been defined to select the relevant instances in the P2P process for the predictive and prescriptive task.

3.1 Business Understanding Steps

The main tasks in this phase are project background analysis, objective determination (business and technical), situation assessment, and project plan creation. The technical objective determination is what distinguishes a PBPM and PrBPM project from other data projects. Its main purpose is to determine which process to monitor, what to predict to realize the business objective, and which prediction type is required (Becker et al., 2020). Moreover, regarding the prescriptive part it is needed to specify the prescriptive task and hence what type of actions/interventions need to be recommended and when in order to reach the business goals. This phase was executed by conducting an interview with the supply chain manager at Company X about the company's operations, P2P process and corresponding KPIs. The questions and a summary of the answers are provided in Appendix D. Moreover, the expediting process at Company X was explored as well as their currently used expediting framework for prioritizing PO items.

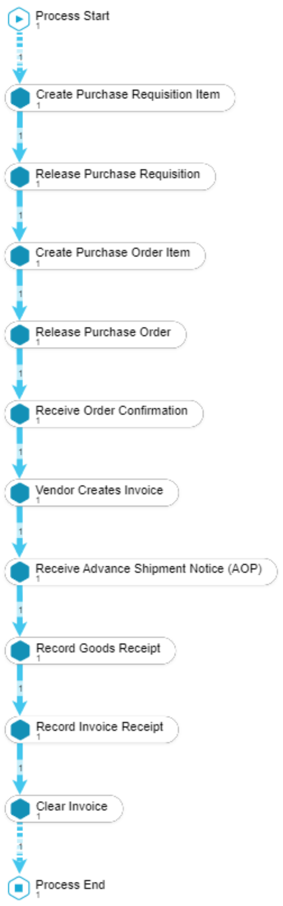
3.2 Data Understanding Steps

The second phase deals with the event log corresponding to the monitored process, in our case the P2P process. The primary goal of the data understanding is to collect the data and get acquainted with it. The P2P data of Company X was made available in process mining platform Celonis, including the basic event log and many other attributes related to purchase orders. First, the basic statistics of the log were checked such as number of cases, events, and activities. Next, it was investigated which other data tables related to the P2P process exist and which attributes are included. Thereafter, the meaning of all the attributes and their values was investigated and was clarified, especially the attributes relevant for prediction. Moreover, limitations with respect to the available and relevant data were identified. This was done by having an interview and detailed data discussion with the process mining expert. After the available data was collected and clarified, possible input features were explored for both the prediction and prescriptive task. While examining the data regarding the prediction task, the prediction type was kept in mind, since different prediction types require different information. For example, since the prediction task for this project is to predict the delivery date, which is a time value, it was checked whether duration information was directly available or could be retrieved from other attributes. Furthermore, it was investigated which data attributes, other than the predicted delivery date, were relevant for the prescriptive task. Therefore, the used data attributes were investigated within the already existing expedite dashboard of Company X within Celonis. The various attributes were further clarified through conversations with the supply chain manager and process mining experts. Furthermore, the filters were defined to put on the P2P process data in order to select the useful data for the prediction and prescriptive task. The values for the filters used to make the selection were defined by having a discussion with the supply chain manager and process mining expert. This was one of the topics brought up in the interview with the supply chain manager (see Appendix D). However, successive iterations were needed to further specify the selection.

The upcoming sections discuss the results for the explained business and data understanding steps, including the P2P process, expedite process, as well as the available data and its limitations.

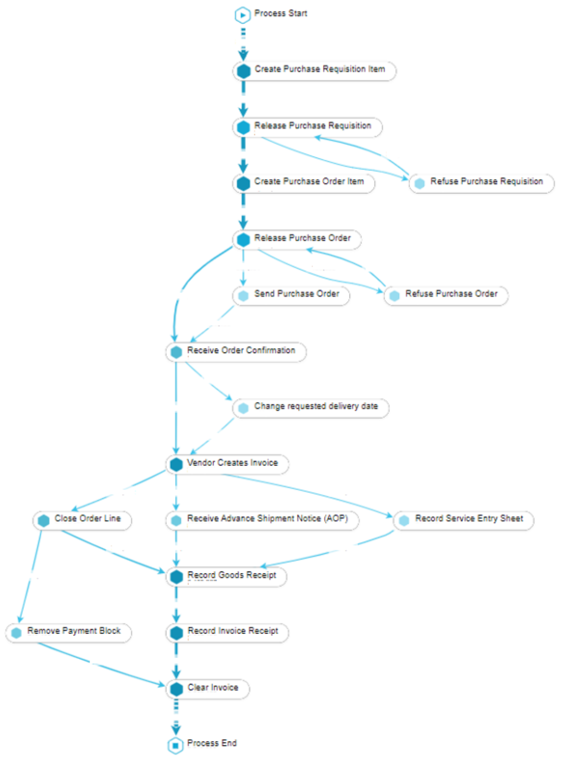
3.3 P2P Process

The P2P process is the main component of a supply procedure, and consists of an integrated set of actions taken to fulfill a requirement for goods or services in a timely manner at a reasonable price. It involves a sequence of steps ranging from creating a purchase order to goods delivery and clearing an invoice (Rzad et al., 2019). According to the supply chain manager, the preferred P2P process, i.e. happy flow, at Company X is the sequence of the following steps: (1) create purchase request, (2) release purchase request, (3) create purchase order, (4) record goods receipt, (5) receive invoice, and (6) clear invoice. It was indicated that the above steps are the main activities and that some other steps are possible in between "create purchase order" and "record goods receipt". Unfortunately, no formal process model is available. When mining the event data in the existing Celonis environment for Company X, the main process variant is shown. This main process variant for a case is depicted in Figure 6. While selecting more activities, it is seen that more options are possible for the execution of the process, since some refuse and change activities have appeared. This process flow is shown in Figure 7 (note that the figure is anonymized by removing the case frequencies normally shown on the arrows and events). However, these newly appearing activities do not belong to the happy flow.



Process Start
Create Purchase Requisition Item
Release Purchase Requisition
Create Purchase Order Item
Release Purchase Order
Receive Order Confirmation
Vendor Creates Invoice
Receive Advance Shipment Notice (AOP)
Record Goods Receipt
Record Invoice Receipt
Clear Invoice
Process End

Figure 6: Process flow with 77.9% of activities



Process Start
Create Purchase Requisition Item
Release Purchase Requisition
Create Purchase Order Item
Refuse Purchase Requisition
Release Purchase Order
Send Purchase Order
Refuse Purchase Order
Receive Order Confirmation
Change requested delivery date
Vendor Creates Invoice
Close Order Line
Receive Advance Shipment Notice (AOP)
Record Service Entry Sheet
Record Goods Receipt
Remove Payment Block
Record Invoice Receipt
Clear Invoice
Process End

Figure 7: Process flow with 95.3% of activities

In order to get a better understanding of the process, the activities in the main process flow (Figure 6) are explained in Table 5. The cases in the P2P process for which these steps are performed are the PO items.

Table 5: Process step description

Number	Process step	Explanation
1	Create Purchase Requisition Item	An employee of an internal department makes a request for the purchase of a specific item.
2	Release Purchase Requisition	The purchase requisition containing multiple requested items and in which the purchase requisition item is included which was created in the previous step, is approved and pushed through the system.
3	Create Purchase Order Item	The approved requested item is added to an existing purchasing order (containing multiple items ordered from the same vendor) where the specific procured material, order quantity, etc. are denoted for the item.
4	Release Purchase Order	The purchase order in which the item is included is approved and sent to the selected supplier.
5	Receive Order Confirmation	A confirmation is received of the placed order from the supplier.
6	Vendor creates invoice	The supplier creates an invoice for the purchase order item and is received by Company X.
7	Receive Advance Shipment Notice	A note with the expected shipping date for the ordered item is sent by the supplier. This is just an estimate of when will be shipped and it does not have to be the case that the shipment has actually started. However, according to the supply chain manager, the previous event 'Vendor creates invoice' is a good proxy of when the item is shipped on the supplier side. When this assumption is made, it could additionally be assumed that when the advanced shipment notice is received the item is being shipped.
8	Record Goods Receipt	The item from the supplier is received on site.
9	Record Invoice Receipt	The invoice of purchased item is booked in the system.
10	Clear Invoice	The invoice is being paid.

The demand for purchase requests is originating from three areas: 40% of the purchasing requests concerns items needed to support maintenance operations, 15% concerns items for inventory replenishment and balancing, and 45% concerns ad hoc requests.
It is seen that the main bottlenecks in the process are the activities (1) creation of PO item, since this is partially still done manually causing the process to take longer, and (2) record goods receipt, due to items being actually delivered too late at a facility *or* the delivery of items is registered too late in the system. These bottlenecks are detected through the observed undesirable KPI values for Purchasing Automation and OTIF rate. These are amongst the most important KPIs for the P2P process which are being monitored:

◇ *Purchasing Automation*: the degree to which activities in the P2P process are being executed automatically.

◇ *Purchase request (PR) to purchase order (PO) cycle time*: the time that has passed from release purchase request to create purchase order.

◇ *Rework*: when change activities are being executed, e.g. changing the price or tax code.

◇ *On Time In Full (OTIF) rate*: rate of items that are delivered on time and in full, e.g. equals 100% when **all** units belonging to a purchase order item are delivered on-time.

As Company X wants to improve the delivery performance within the P2P process, this research focused on this last KPI, the OTIF rate. Therefore, Company X wants to have an estimate of when the PO items will be delivered. Predictive process monitoring techniques can be used to generate these predicted values. This can then be used to act proactively upon PO items that are estimated to be delivered too late.

3.4 Expediting Process

Like explained in Section 1.1, Company X calls their suppliers when purchased items are too late. This process is called expediting. Currently, Company X has an expedite dashboard which shows the prioritized list of purchased items from suppliers that need to be expedited for the day. This priority list is generated as follows (by their original prescriptive model). First, within the expedite dashboard, the items are being selected that are already overdue. Subsequently, the expedite score is being calculated for those items using a weighted scoring model. In any case, the item is either deleted or not yet overdue, no expedite score is calculated. Eventually, the items with the highest expedite score will get the highest priority for expediting.

The expedite score, indicating the priority for expediting a PO item, is calculated by a weighted scoring model. This expedite score is based upon the following factors: (1) material criticality, (2) stock versus non-stock, (3) supplier historical delivery performance, and (4) days overdue. The value for the expedite score for an item ranges between 0 and 10, where a higher value denotes a higher priority for expediting. This score is a weighted sum of the scores for the four factors (values $\in [0, 10]$), with material criticality and stock versus non-stock accounting each for 30% and supplier historical delivery performance and days overdue each for 20%. In the following, the four factors are explained.

1. *Material Criticality*: This factor indicates how important the product is for Company X to keep all operations up and running. For example, a PO item with the highest priority (value C1 or 1) is essential to prevent bottlenecks in the core processes. If the PO item has a material criticality of C1 or 1, a maximum score of 10 is assigned. If the criticality is C2 or 2, a score of 6 is assigned. Whenever, the criticality is C3 or 3, a score of 4 is noted. Given that the criticality is of a lower level, a score of 2 is assigned.

2. *Stock versus Non-stock*: If the item is an item for which there is supposed to be a stock, and stock on hand is zero a maximum score of 10 is assigned, otherwise 5.

3. *Supplier Historical Delivery Performance*: This factor looks at the OTIF of a vendor of last month and compares it to the OTIF of last year. If the OTIF of last month is 10% points lower than the OTIF of last year a maximum score of 10 is assigned, otherwise 0.

4. *Days Overdue*: This is the difference in days between the date on which the score is calculated, "Today", and the delivery date as planned, which is either the contractual date or as last updated by the vendor. It is then calculated by subtracting the delivery date as planned from the date of "Today". For each 2 days the PO item is overdue, the score increases with 1 point, to a maximum of 10 points.

The expedite score is then calculated by the following formula: 0.30 $*$ Material Criticality Score + 0.30 $*$ Stock versus Non-stock Score + 0.20 $*$ Days Overdue Score + 0.20 $*$ Supplier Historical Delivery Performance Score.

During earlier conversations with the supply chain manager it was discovered that a recommender type of prescriptive system is demanded by Company X. To be precise, a system recommending for a running case (a not fully delivered PO item) the priority it should get for expediting, and overall generating a prioritized list of purchased items from suppliers that need to be expedited for the day. Together with Company X it was decided to build a prescriptive model based on the just explained existing framework at Company X for expediting.

3.5 Available Data

Data regarding the P2P process and especially delivery performance has been measured for the past five years. This data is embedded into process mining platform Celonis for year 2019 until 2022 and is sourced from their SAP S4HANA system and SAP ECC system. The source data within these systems is updated on a daily basis, and therefore the data visible in Celonis as well. The next paragraph explains the available and relevant data tables regarding the P2P process in more detail.

3.5.1 Data Tables

Different data tables related to the P2P process exist. Table 6 describes the most relevant data tables available including their most relevant attributes.

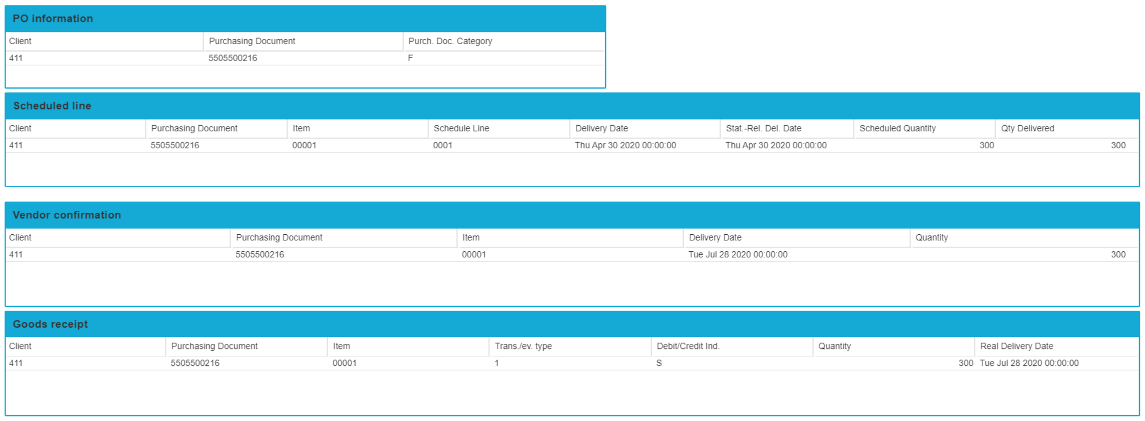
Table 6: Data tables P2P process

Official table name	Table description	Data fields
Purchase requisition	Contains data about a PR. This is a request to purchase a certain quantity of a material such that it is available at a certain point in time. It is an internal document (LeanX, 2022a).	client, PR number, PR item number, purchasing document number, purchasing document item number, purchasing group, deletion indicator, processing status (e.g. PO created, scheduling agreement created), creation indicator (e.g. production order, real-time, material requirements planning), purchasing group, resource (created by), requisitioner, material number, material group, plant, quantity requested, desired vendor number, requisition date, delivery date, PO date, release date
Purchasing document	This is the header table and contains information about the whole document, and thus for one or more items. For example, information about the vendor and the document number is contained in the document header, whereas the material description and the order quantity are specified in each item. The purchasing document can be a request for quotation, quotation, PO, contract, or scheduling agreement (LeanX, 2022b). The table will be mainly used to get information on the PO. This is a buying request to an external supplier to supply certain materials, formalizing a purchase transaction.	client, purchasing document number, purchasing document category (value F indicates a PO), company code, purchasing organisation, creation date, created by (resource at Company X), purchasing group, currency, vendor number, status (e.g. third-party order from CRM, returns order from incorrect delivery, enjoy puchase order), deletion indicator, delivery completion indicator
Purchasing document item	Contains information for a specific item within a purchasing document. This table can be used to get detailed information on the items within a PO.	client, purchasing document number, purchasing document item number, company code, plant, storage location, deletion indicator, delivery completion indicator, material number, material group, order quantity, net price, net order value, PR number, PR item number, planned delivery time (the lead time for an item as determined by Company X and which is given as limit to the vendor), short text (could contain type of material used)
Scheduling agreement scheduled lines	Contains the scheduled lines for all purchasing document categories (request for quotation, quotation, PO, contract, or scheduling agreement). A scheduled line for a PO item contains the information about a specific delivery moment for a specific quantity of the purchased item.	client, purchasing document number, purchasing document item number, scheduled line number, item delivery date (is adjustable and is the expected delivery date), contractual (planned) delivery date, scheduled quantity, quantity delivered
Vendor master	Contains general data about the external supplier.	client, name, vendor number, country, city
Vendor confirmations	Contains information about the vendor confirmations. A vendor confirmation gives information on the purchased item as confirmed by the supplier, e.g. when the supplier expects to deliver the item.	client, purchasing document number, purchasing document item number, delivery date (as confirmed by the vendor), quantity as per vendor confirmation
History per purchasing document	Contains information on what happened in the past with a purchasing document. Information on past events for purchasing documents are recorded. Example event types are goods receipt, invoice receipt, stock transfer, and down payment clearing. Especially the goods receipts event type (actual deliveries) for a PO item is relevant for this research.	client, purchasing document number, purchasing document item number, transaction/event type (value 1 for goods receipt), debit/credit indicator (value S for confirmed, value H for canceled), quantity, date of event/transaction (e.g. of goods receipt event), delivery completion indicator, material number
Reservation requirements	This is a table related to inventory management and contains reservation information for items. It shows for example the quantity that is required of an item of a specific material and the quantity of an item that has already been claimed.	client, purchasing document number, purchasing document item number, material number, plant, storage location, date when requested quantity is required, reserved (required) quantity, issued quantity (quantity that is already claimed)
Storage location data for material	Contains material information at storage location level, e.g. the available stock.	client, material number, plant, storage location, quantity in stock at location
Activity table (P2P activities)	It shows for each case the events that happened and their timestamps. The cases are on item level.	case ID, client, purchasing document number, purchasing document item number, activity, event time, user name (resource)

For sake of clarity, the attributes client, company code, purchasing organization, purchasing group, and vendor are explained. First, the internal attributes, thus concerning Company X, are explained. The value for the attribute 'client' indicates the business unit of Company X for which the purchase is done. Next, such a business unit consists of multiple companies which are represented by values for the attributes 'company code' and 'purchasing organization'. Finally, within these companies different purchasing groups exist to which employees are assigned and allowed to purchase specific items. Lastly, the vendor is an attribute which is external since its value indicates the external supplier selected for delivery.

The goal is to create an event log from the available data for PO items, including relevant features for the predictive as well as prescriptive task in this thesis. As can be seen in Table 6, certain attributes are overlapping for the various data tables. These attributes can then be used to connect information from within separate tables for a PO item such that the event log with all relevant features can be created. PO item information from different tables can be linked through looking at whether a line in a table is for the same client, purchasing document, and purchasing item.
To further understand the information available for PO items, how the information from different tables is related, and what data is relevant for the prediction as well as prescriptive task, data was combined from the various tables in multiple analyses. Figure 8 shows such a combined analysis that was created as an example. Information about a PO item is combined from the scheduling agreement scheduled lines table, the vendor confirmations table and history per purchasing document table. For the specific client, purchasing document number, and purchasing document item number it is seen that a scheduled line, vendor confirmation, and goods receipt event are present. A scheduled line contains the information about a specific delivery moment for a specific quantity of the purchased item. It is, for example, possible to have multiple scheduled lines for one PO item. This means multiple delivery moments exist on which parts of the entire ordered quantity are delivered for the PO item. The next subsection, 'data limitations', elaborates on how to deal with the multiple delivery moments. According to the scheduled line a delivery of 300 units for item

00001 was planned for the 30th of April in 2020, and the full scheduled quantity was eventually delivered. In the vendor confirmation, it can be seen that the supplier indicates that the goods will be delivered the 28th of July. Then, the history of purchasing document table shows that the actual delivery took place the 28th of July since that is when the goods receipt event took place for a quantity of 300 units for the PO item.


PO information

Client	Purchasing Document	Purch. Doc. Category
411	5505500216	F

Scheduled line

Client	Purchasing Document	Item	Schedule Line	Delivery Date	Stat.-Rel. Del. Date	Scheduled Quantity	Qty Delivered
411	5505500216	00001	0001	Thu Apr 30 2020 00:00:00	Thu Apr 30 2020 00:00:00	300	300

Vendor confirmation

Client	Purchasing Document	Item	Delivery Date	Quantity
411	5505500216	00001	Tue Jul 28 2020 00:00:00	300

Goods receipt

Client	Purchasing Document	Item	Trans./ev. type	Debit/Credit Ind.	Quantity	Real Delivery Date
411	5505500216	00001	1	S	300	Tue Jul 28 2020 00:00:00

Figure 8: Data table analysis

Moreover, within another analysis it is observed that material and stock information can be linked for a PO item by looking at whether a line in the purchasing document item table has the same attribute values for client, material number, plant, and storage location as lines within the reservation requirements and storage location tables. This is then relevant for the prescriptive task, since the expedite framework (see Section 3.4) uses stock information among others to determine the priority for expediting regarding a PO item.

Furthermore, in order to build the event log, the activity table is very important as it contains the standard event log attributes: case ID, activity, and timestamp. Moreover, it contains the activities (and their timestamp) that go along with the change of a data field in a certain table. These change activities are observed for the following tables: vendor confirmations, scheduled agreement scheduled lines, purchase document, purchase request, and purchase document item. For example, such a change activity in the vendor confirmations table shows when the delivery date as confirmed by the vendor is changed. Nevertheless, this happens only for 0.0003% of all cases. Moreover, for 12% of all cases the requested delivery date (delivery date attribute) is changed in the scheduling agreement scheduled lines table. Another example is that for 86% of all cases change activities occur where a data field of the purchase document item table is changed. It is noted that the data field that is being changed is one of the following: delivery completion indicator, final invoice, net order price, deletion indicator, tax code, storage location, order quantity, order acknowledgement, outline agreement, principal agreement, supplier material number, or manufacturer part number. Furthermore, for the purchase requisition table change activities 'release PR' and 'refuse PR' are found. Lastly, for the purchase document table the release indicator is sometimes changed when activity 'release PO' or 'change approval for PO' occurs.

The next paragraph explains the limitations regarding the available data. Thereafter, Section 3.5.3 elaborates on the available attributes that are most relevant for the prediction task as well as for the prescriptive task.

3.5.2 Data Limitations

Multiple limitations exist with respect to the available data according to the process mining expert. One of these limitations is regards the data with respect to the delivery date of a purchased item. Three main data fields are related to the delivery date, which are (1) the contractual delivery date in the scheduling agreement scheduled lines table, (2) the delivery date in the vendor confirmations table (date for which the vendor announces that the ordered item should be delivered), and (3) the date of a goods receipt event in the history per purchasing document table (the actual delivery date). It seems to be the case that the delivery date as confirmed by the vendor is often not filled in beforehand by the vendor, and is often filled in simultaneously with the date of goods receipt. Therefore, it seems more reliable to take the date of the goods receipt event happening as the actual delivery date instead of the date as confirmed by the vendor. However, for example, for a specific vendor to which will be referred to as “supplier Z”, it is seen that the delivery date as confirmed by the vendor is well filled in beforehand. But, in their case the goods receipt event date is then often recorded too late, while the goods were already there. As this is known for this specific vendor, it would be an option to only for this vendor use the delivery date as confirmed by the vendor as the actual delivery date. Nevertheless, according to the process mining expert and supply chain manager it is more reliable to take the date of goods receipt event as the actual delivery date for all vendors. Therefore, only the date of goods receipt will be used in the analysis.
Another interesting observation is that, for some purchased items, a certain quantity is delivered at one point in time, while the remaining quantity is delivered at another point in time, resulting in partial deliveries. This is seen in the event log by having multiple goods receipt events for one purchased item. It is a remarkable situation as in such a case only one scheduled line was found, meaning there was only one scheduled delivery moment for the entire quantity. While analyzing the data in Celonis, it is observed that this happens for 10% of the purchased items. Sometimes, the same phenomenon can be detected by finding multiple vendor confirmation lines for one purchased item, indicating multiple delivery moments are planned to take place according to the vendor in order to fulfill the ordered quantity. The existence of partial deliveries needs to be taken into account when developing the prediction model. According to the supply chain manager, these partial deliveries should be handled as follows: for the purchased items for which multiple delivery moments exist, only the date of the last goods receipt event (and the quantity delivered up until that point in time) should be used when training the prediction model. The reason is that the date should be predicted for which all units of a purchased item are delivered, since this is most relevant for the OTIF.
The final limitation has to do with the data attribute ‘material number’ that is not well-filled. It is seen that many values are missing. However, for these cases occasionally a short text comment is given as attribute value for the attribute “short text”, indicating the type of material/material number. Moreover, it might also happen that a ‘part number’ is available instead, which indicates the type of material as specified by the vendor. This should be taken into account when the attribute ‘material number’ is used as input for the prediction model, since pre-processing is most likely required in order to be able to use the data.

3.5.3 Possible input features

Together with Company X, a list with potential input features for the predictive models was created. This list is given in Table 7. Besides, the table indicates whether the feature is directly available as data attribute in Celonis as well as whether and how it needs to be engineered from other data attributes.

Table 7: Possible input features predictive models

Data attribute	Type of attribute	Engineering
Item number	case	-
Business unit (BU)	case	-
Vendor	case	-
Plant	case	-
Material number	case	-
Last year OTIF	case	-
Last month OTIF	case	-
PO lines per vendor	case	Calculating the count of PO items that exist for each vendor
Number of changes done to the delivery date	case	Calculating how often the activity 'change requested delivery date' has occurred for each PO
Number of days between PO creation and the contractual delivery date	case	Calculating the date differences in days between the contractual delivery date and the date of PO creation
Activity	event	-
Time since last event	event	Compute difference in event time between all event pairs in the cases
Time since case start	event	Compute difference in event time between each event and the first event of the case
Time since midnight	event	Sum of all hours, minutes, and seconds that have passed since midnight for each event in the cases.
Weekday	event	Transform event time to day of the week.
Target: Days between contractual delivery date and goods receipt event (actual delivery date)	case	Calculate the number of days between the last goods receipt event and the contractual delivery date.

The selected relevant attributes for the prescriptive model are based on the already existing expedite framework of Company X, which was explained in Section 3.4. Table 8 lists the possible input features. These attributes are most relevant since they are crucial for the calculation of the expedite score and generation of the expedite priority list as in the current expedite framework of Company X.

Table 8: Possible input features for the prescriptive model

Data attribute	Type of attribute	Engineering
Material criticality	case	-
Stock vs non-stock	case	-
Days overdue score	case	Calculated by checking the value for the number of days between "Today" and the planned delivery date. Subsequently the days overdue score is calculated based on the logic as explained in Section 3.4.
Last month OTIF	case	-
Last year OTIF	case	-

It can be noted that the OTIF of last year and OTIF of last month are selected for both the predictive models and prescriptive model. Although the attributes are the same, different values for these attributes will be provided as input to the predictive models than to the prescriptive model. This is because different process instances (PO items) are selected to be given as input to the two types of models, due to using different process filters. The following section elaborates further on these process filters.

3.6 Process filters

Lastly, filters were defined to put on the P2P process, together with the supply chain manager and process mining expert, to be able to select the required data for the predictive as well as prescriptive task, e.g. certain kinds of instances, a part of the process, or particular types of events (Becker et al., 2020).

Firstly, the filters for the prescriptive task are discussed. Different process filters are needed for the training phase of the predictive model than for the deployment phase, because different data is required: (1) data used for training the model, and (2) data used when the model is implemented and for which the predictions will be made. The resulting filters that are used to select the appropriate process instances for these two purposes are summarized in Table 9.

Table 9: Process filters for model training and implementation

Training	**Implementation**
Delivery is completed	Delivery is not completed
Deletion indicator has value 'not deleted'	Deletion indicator has value 'not deleted'
The order type should be 'goods' (so no services are included for example)	The order type should be 'goods'
Maximum number of events per case is 50	Maximum number of events per case is 50
Include only external trading partner	Include only external trading partner
Include only F, K, blank, and O values for account indicator	Include only F, K, blank, and O values for account indicator
Exclude cases going through the activity 'reopen order line'	Exclude cases going through the activity 'reopen order line'
Include all cases that flow through create purchase order item, and at least gone through record goods receipt. And crop the process until the last goods receipt event.	-
When a case flows through a cancel goods receipt, end the case at first goods receipt event. This is not taken into account as an actual filter but in the computation of the target value, the actual delivery date has been changed for these cases to the first goods receipt event instead of the last goods receipt event.	-
Include cases where difference between actual delivery and contractual date is less than 90 days	-
-	The case should not be fully invoiced
-	The contractual delivery date should not be blank

After the training filters are applied on the event log around 63% of all cases (2.3 Million cases) remained. In order to further refine the event log and to prevent the log containing outlier/highly infrequent traces, the choice has been made to include only the 55 most frequent process variants. These variants had a case coverage of $> 0.20\%$. Thereafter, only 36% of all cases was left, coming down to 1.4 Million cases. The characteristics of the finally selected cases for predictive model training are summarized in Table 10.

Table 10: Event log characteristics when process filters have been applied

Characteristic	Value
Number of traces	1,413,994
Total number of events	9,765,979
Number of unique activities	13
Minimum trace length	4
Maximum trace length	9
Average trace length	7

Although the filter of a maximum of 50 events per case was applied, the maximum number of events in the obtained log is much lower. Apparently, all the filters together have eliminated traces with a trace length longer than 9 events. The average trace length of traces in the log is 7. Therefore, it is evident that the total number of events in the log equals approximately 7 times the number of traces in the log.

Next, the filters regarding the prescriptive model are discussed. A distinction has been made between three types of filters: (1) filters to put on the data used for evaluating the model on past expedited items, (2) filters to put on the data used for evaluating the model on items that will be expedited in the future, and (3) filters to put on the data when the model is implemented. These filters are listed in Table 11. The underlying method for the historical analysis, future analysis, and model implementation are explained in detail in Section 6.1.3 within Chapter 6.

Table 11: Process filters to select the cases used in the prescriptive model

Historical Analysis	**Future Analysis**	**Implementation**
	Include only PO items with the highest material criticality: C1 or 1	
Include only PO items for which there is a predicted delivery date		
The PO was created in time window [February 2022, September 2022]		
The order line was overdue (at some point "Today" was later than the planned delivery date) and expedited	The order line is predicted to be overdue in the upcoming n days and will be more than z days late	The order line is currently overdue OR is predicted to be overdue in the upcoming n days and will be more than z days late
	The delivery for the PO item is not yet completed	The delivery for the PO item is not yet completed
The PO item is not deleted	The PO item is not deleted	The PO item is not deleted
The PO document is not deleted	The PO document is not deleted	The PO document is not deleted
Only PO items of order type 'goods' are included	Only PO items of order type 'goods' are included	Only PO items of order type 'goods' are included

The two variables n and z can be changed by Company X. During discussions with the process mining expert and supply chain manager it was decided that these variables will initially be given the following values: $n = 14$ and $z = 1$. Moreover, for the future analysis and implementation purpose it is not needed to filter on the availability of the predicted delivery date. Since the predictive model will be implemented upon that point, there will always be a prediction for the PO items. Furthermore, for the historical analysis, only data starting from February 2022 was reliable to use according to the process mining expert, since the expedite team only really started recording their expedite actions from that point in time.

The next chapter elaborates on the second part of the data understanding phase where an in-depth data analysis has been performed to get a deeper understanding of the possible input features and P2P process characteristics.

4 Data Analysis

The aim of this chapter is to get a thorough understanding of the process characteristics and possible input features for the predictive models as well as prescriptive model. Therefore, an in-depth data analysis is performed. This data analysis and also second part of the data understanding phase is important, since it can identify whether certain preprocessing steps need to be performed on the possible input features and whether the P2P process behavior is suitable for the use of specific predictive models. First, the executed steps are described, after which the results of the analysis are presented.

4.1 Data Analysis Steps

First of all, the P2P process was further investigated to understand whether it contains interesting control-flow relations other than seen in Figures 6 and 7, such as concurrency. This is important to know, since this thesis aims at comparing graph neural networks with other DL-based PBPM methods for predicting the delivery date. Graph neural networks exploit the structural properties found in process models, like loops and parallelisms, through which predictive performance is expected to improve. Then, in order to do a relevant comparison between graph neural networks and other methods, the P2P process should contain some interesting structural properties, otherwise there is no benefit in using this network. To visualize this possible behavior, process models were mined using various process discovery algorithms. The process models are an abstract representation of the information in the event log (Aalst et al., 2011). The event log remaining after the application of the process filters for model training (see Table 9) was used to mine the process models. Process mining tool ProM and Python library PM4PY were used in order to run different mining algorithms (Fraunhofer FIT, 2021b; Van Dongen et al., 2005). Four different process mining algorithms were used to create a Petri Net representation as process model from the event log: the (1) Alpha Miner (AM), (2) Heuristics Miner (HM), (3) Inductive Miner (IM), and the (4) Inductive Miner infrequent (IMf). The Alpha Miner mines a process model based on analyzing the ordering relations between activities in the event log. The Heuristics Miner is a robust and heuristics-based method for process discovery. It can discover short loops, but it is not capable of detecting non-local, non-free–choice constructs (Naderifar et al., 2019). The Inductive Miner splits the event log and constructs the model recursively. An extension of this miner is the Inductive Miner infrequent, which is better at handling infrequent behavior in the process (Leemans et al., 2013). The process models discovered by the algorithms were evaluated in terms of fitness, precision, and generalization. These measures to evaluate a process model are commonly used in previous research (Aalst et al., 2011; Leemans et al., 2013). These three quality criteria are important to check, since an unfitting model cannot reproduce all behaviour seen in the log, whereas an imprecise model allows for too much additional behaviour that is not described in the log. Moreover, a model that is not general can only describe the behaviour in the log and thus might be able to reproduce future behaviour absent in the log (Leemans et al., 2013).

Furthermore, the data quality had to be checked for the potential input features for the prescriptive models as well as predictive model. Issues like spelling and format mistakes, invalid attribute values, missing values, as well as outliers had to be checked (Welcker et al., 2012). Invalid attribute values should be judged by expert knowledge, thus when suspicions arose, these were checked with the process mining expert. In order to thoroughly examine the available data for the possible input features an exploratory data analysis (EDA) was conducted. This analysis consists of three main steps: (1) the univariate analysis, focusing on the descriptive statistics of the data attributes as well as missing value and outlier detection, (2) the bivariate analysis, looking at the relationships between pairs of data attributes, and (3) feature selection, highlighting potential impactful data attributes to be used as input into the predictive model (Komorowski et al., 2016). All three steps were performed for the features regarding the predictive part. However, for the features regarding the prescriptive part, only the univariate analysis was performed. Those input features are solely used to select the PO items for expediting and to calculate an expedite score for prioritizing the

items, as explained in Section 3.4. Therefore, there is no target variable like for the predictive part, and thus the relation between the features and target does not need to be investigated.
The univariate analysis focused on the characteristics of the data for each attribute separately (Visual Design, 2021). Descriptive statistics such as mean and median values, as well as outlier and missing attribute values were acquired by analyzing the selected data in Python. The outliers were detected by inspecting boxplots and by using the z-score. This score indicates how many standard deviations a value for an attribute is away from the mean value for that attribute (Ektamaini, 2020). It was chosen to classify a value as outlier if it is more than 2 standard deviations away from the mean.
Next, for the predictive part a bivariate analysis was performed to understand the relation between attributes and the target variable (Shixin, 2020). Different statistical and visual methods exist to do such an analysis. The main method used to inspect the relationship between the categorical attributes and numerical target was the one-way Analysis of Variance (ANOVA) of type 2. The type 2 ANOVA was used since it is suggested to be better in the presence of unbalanced data (Langsrud, 2003). Besides, one-way means that there is only one independent variable for which the relation is tested with the dependent variable (Glen, 2022a). In our case, one categorical variable with the target variable. The ANOVA method calculated an F-score and p-value for each categorical feature. A p-value below or equal to 0.05 implies a significant difference in mean target value between at least two of the categories of the feature (Glen, 2022a). The F-score is the ratio of the variation between the mean target values of the different categories and the variation of target values within the different categories. Therefore, a higher F-score indicates a stronger relation between the categorical feature and target (Zach, 2021). The analysis was performed on a prefix-level (event-level), not on a case-level. The underlying reason is that the predictive models also get prefix-level input, so in order to get an estimation of the effect of a categorical variable on the predicted outcome it seemed fair to represent the input similarly. For analyzing the relationship between a numerical attribute and numerical target a correlation matrix was used, which indicates the strength of linear relationship between the variables (Muthukrishnan, 2021). The relation was as well be visualized by a scatterplot for each of the numerical features.
Subsequently, for the predictive part, four automatic feature selection methods were tested to explore which features are likely to have a high impact on the target variable, the number of days between the actual delivery date and contractual delivery date. All four methods train a predictor which predicts the target variable and discovers which features are most influential on the predicted value. The first method is a Random Forest Regressor, to which the select from model technique of the Sklearn feature selection method was applied. This method selects features based on importance weights (scikit-learn developers, 2022b). The second and third method also use the select from model feature selection technique from Sklearn, but apply this to a Ridge Regressor with cross-validation and Lasso Regressor with cross-validation. Lastly, the sequential feature selector (forward selection) of Sklearn was applied to the Ridge Regressor with cross-validation. It adds features to form a feature set in a greedy manner. At each stage, this estimator chooses the best feature to add based on the cross-validation score of the Ridge Regressor (scikit-learn developers, 2022c).

The upcoming sections elaborate on the in-depth analysis of the structural properties of the P2P process as well as the exploratory data analysis of the input features for the predictive models and prescriptive model.

4.2 Process models

Like explained in the previous section, four different process mining algorithms have been used to create a Petri Net representation from the filtered event log. In order to analyze the structural properties of the process, the process model that best represents the process needs to be selected. Therefore, the resulting Petri nets for each of these algorithms are analyzed using the evaluation criteria fitness, precision, and generalization. The results for the different models on these criteria are listed in Table 12.

Table 12: Results for the discovered process models

Miner	Fitness	Precision	Generalization
AM	0.650	1.000	0.998
HM	0.820	0.950	0.793
IM	1.000	0.340	0.998
IMf (with 20% of noise filtered out)	0.970	0.530	0.998
IMf (with 40% of noise filtered out)	0.950	0.730	0.998

The Petri net generated by the AM has the highest precision value, and even the maximum value possible for precision. This means that all the behaviour that the model allows for is seen in the log as well. However, the fitness value is lowest compared to the other models. Hence, the model cannot reproduce all behaviour described in the log. The fitness value is highest and the maximum for the process model generated by the IM. Nevertheless, an extremely low precision value of 0.340 is observed. In consequence, only the models created by the HM, IMf with 20% of noise filtered out, and IMf with 40% of noise filtered out have a chance of being the best model. The model discovered with the IMf where 40% of noise filtered out is a better choice compared to the model discovered with the same algorithm but filtering 20% of the noise. By increasing the noise filtering up to 40% more infrequent behavior is removed. This increases the precision value with 0.200 and decreases the fitness value with only 0.020. The value for generalization has not changed and is exceptionally high with 0.998. The generalization value for the model resulting from the HM, 0.793, is fairly low compared to that. But, the its value for precision is much better with 0.950, and the fitness value is as well not much worse with 0.820. Therefore, those two models are additionally evaluated on simplicity. Regarding this quality criterion, the simplest model that can explain the behavior contained in the log is the best model (Aalst et al., 2011). The simplicity of a model can be assessed by looking at the number of places, transitions and arcs necessary to express its behaviour (Leemans et al., 2013).

Table 13: Simplicity of models created by the HM and IMf with 40% of the noise filtered out

Miner	#Places	#Transitions	#Arcs
HM	33	39	106
IMf (with 40% of noise filtered out)	19	23	50

Hence, when looking at the Petri nets in Figure 35 and 36 in Appendix A and their number of places, transitions, and arcs in Table 13, it can be concluded that the process model discovered by the IMf with 40% of the noise filtered out best represents the P2P process. When further investigating the process model in Figure 36 in Appendix A, it is observed that the process includes various interesting structural properties. For example, there are two parts in the process where activities occur in parallel, and where in the parallel flows also choices are present. Moreover, there are four other moments where a choice can be made to either do a certain activity or skip the activity. Therefore, the process is definitely interesting enough to be used as input for the graph neural networks.

4.3 Exploratory Data Analysis

This section elaborates on the exploratory data analysis performed on the possible input features for the predictive models as well as prescriptive model. First, the univariate and bivariate analysis are described for the predictive part of this thesis. Next, the detected preprocessing steps are explained that have to be performed during the data preparation phase of the predictive process monitoring. Subsequently, the results from the four feature selection methods are discussed. Thereafter, the univariate analysis is presented for the prescriptive part of this thesis.

4.3.1 Predictive Part - Univariate Analysis

In the upcoming paragraphs, the results are described of the univariate analysis performed on the attributes that serve as possible input features for the predictive models. The data for the possible input features is retrieved for the cases in the P2P process selected by the process filters for model training (see Table 9). The focus is on the descriptive statistics of the data attributes, missing values for those attributes, as well as outlier values. The descriptive statistics and outliers have been examined for the numerical attributes and categorical attributes separately.

Missing Values
In the filtered log, most missing values appeared for the attribute material number with around 24% of all data. Therefore, possible substitutes were inspected: material group and a combination of material number, part number, and short text. The combined attributes is constructed as follows. It takes the value of material number, but in case of a missing value it is replaced by the part number, and if no part number is available, the missing value is replaced by the short text description for the material. This combined substitute feature was created together with the process mining expert and supply chain manager of Company X, and will be referred to as material/part number. Subsequently, missing values are detected for material group, material/part number, OTIF of last year, and OTIF of last month. The exact numbers and percentages of missing values for these attributes are given in Table 14.

Table 14: Missing values for the various attributes

Attribute	Number of missing values	Percentage of missing values	Number of cases with missing value	Percentage of cases with missing value
Material/part number	8	8.192e-05	1	7.072e-05
Material group	254	0.003	34	0.002
OTIF last year	393931	4.034	58522	4.139
OTIF last month	936724	9.592	137001	9.689

Overall, when only considering the attributes listed in Table 14, 9.619% of all rows in the dataset contain missing values for at least one of these attributes. This percentage is not equal to the sum of the percentages of missing values for each of the attributes, because a row can have missing values for multiple attributes. Section 4.3.3 explains how these missing values are handled.

Numerical Attributes
First, the descriptive statistics are provided in Table 15 for the numerical attributes.

Table 15: Descriptive statistics numerical attributes

Attribute	Mean	Median	Mode	Min	Max	Standard Deviation	Skewness	Kurtosis
OTIF of last year (%)	63.229	64.792	37.737	0	100	21.585	-0.425	-0.495
OTIF of last month (%)	76.342	83.495	100	0	100	22.012	-1.317	1.772
Number of PO lines per vendor	33385.2	10421	164119	0	164119	43721.9	1.589	2.039
Number of changes done to delivery date	0.047	0	0	0	12	0.222	5.486	51.122
Number of days between PO creation and the contractual delivery date	24.719	15	8	-720	3685	36.712	11.49	543.026
Time since last event (seconds)	320579	17570	0	0	5.980e+08	2.377e+06	127.429	21810.1
Time since case start (seconds)	1.066e+06	204157	0	0	5.987e+08	5.715e+06	56.300	3997.740
Time since midnight (seconds)	32803.700	36038	1	0	86399	23776.500	-0.019	-0.991
Target: number of days between actual and contractual delivery date	-5.135	-3	0	-3673	89	33.491	-9.264	452.875

Especially, for PO lines per vendor, number of days between PO creation and contractual delivery

date, time since last event, and time since case start, a difference is observed between the mean and median value. The mean value is much higher than the median for all these attributes. This could indicate that there are outliers in the dataset for these attributes. Another peculiar finding is that the minimum for the number of days between PO creation and contractual delivery date is a negative value. It points to the fact that there are cases in the dataset where the PO is created at a date later than the contractual delivery date. This phenomenon should not be possible, and therefore negative values for this attribute are invalid. Section 4.3.3 explains how is dealt with this phenomenon. Besides, the minimum value for the target is negative. However, this just means that a PO item arrives earlier than the contractual delivery date. But, the minimum value is definitely an outlier since a value of -3673 days means that a PO item arrived more than 10 years before the contractual delivery date, which is unreasonable.
Skewness is a measure of symmetry of the data distribution, whereas kurtosis is a measure of the heaviness of the distribution tails (Gawali, 2021). Multiple baselines are found for the ranges for skewness and kurtosis that indicate whether the distribution can be assumed to be normal. In this thesis a range of [-1,1] for skewness and [-3,3] for kurtosis are applied (Gawali, 2021). Taking these ranges into account, clearly 7 out of the 9 attributes are skewed. Solely OTIF of last year and time since midnight are not skewed. Extreme positive skewness is found for the number of changes done to delivery date, number of days between PO creation and contractual delivery date, time since last event, and time since case start. It means most values for these attrbutes are plotted on the left side of the graph and the tail of the distribution is spreading to the right. Large negative skewness is found for the target variable. This also means the mode is bigger than the median, and the median is higher than the mean. However, before deciding whether to transform the values for these attributes with a log-transformation, the values for skewness should be examined again after preprocessing steps like removing missing and outlier values. Thereafter, the distribution could have changed resulting in a different skewness value. This decision is made in Section 4.3.3. Furthermore, 5 out of the 9 attributes have an extremely high and positive kurtosis value. Their distributions can then be visualized as a thin bell with a high peak. This implies a higher chance of outliers for these attributes (Gawali, 2021).

Furthermore, Table 16 shows the proportions of cases that are delivered later than the contractual delivery date, before the contractual delivery date, and exactly on-time. The proportions in Table 16 are for the raw total dataset.

Table 16: Target - difference in days between actual and contractual delivery date

Category	Percentage of cases
Delivered later than the contractual date	33.657
Delivered before the contractual date	59.179
Delivered on the contractual date	7.165

Clearly, the cases that are being delivered on-time are the majority, accounting for 66.344%. This is not unexpected, since it is normal that items are being delivered on-time mostly and that the cases for which it goes “wrong” are more of an exception.

The outliers have been investigated by looking at boxplots as well as the z-score, like explained Section 4.1. Outliers are found for the following attributes: number of of PO lines at a vendor, number of changes to delivery date, and the target variable. However, for ‘number of changes to delivery date’, a value is already classified as outlier when it is bigger than zero. This is because this attribute takes value zero for 95.407% of the data, meaning that a higher number of changes occurs only rarely. Nevertheless, removing these values, makes the attribute redundant since it will only have one value and does not give information about changes made to the delivery date anymore. Section 4.3.3 explains how is dealt with this observation. Table 17 shows the outlier values for the other attributes as well as the percentage of cases being an outlier for the specific attribute.

Table 17: Outliers for numerical attributes

Attribute	Outlier value (range)	Percentage of outlier prefixes
Number of of PO lines at a vendor	164,119	5.805
Difference in days between PO creation & contractual delivery date	[-720,-49] and [99,3685] given by z-score, but also [-48,0) with domain knowledge	5.850
Time since last event (seconds)	$[5074458, 597974400]$	0.938
Time since case start (seconds)	$[12495252, 598716024]$	0.608
Target variable (days)	$[-3673, -70]$ and $[60, 89]$	11.029

Overall, the percentages of outliers for the various attributes are not particularly high, with a maximum of 11.029% for the target variable. An additional range of outliers has been added besides the ranges resulting from the z-score for the attribute difference in days between PO creation and contractual delivery date. Like mentioned before, it should not be possible to have cases in the dataset where the date of PO creation is later than the contractual delivery date. Therefore, the remaining and thus invalid negative values ([-48,0)) are added to the list of outliers.

The following paragraph elaborates on the descriptive statistics and distributions for the categorical attributes.

Categorical attributes
First of all, the cardinality is examined for the categorical attributes. This describes the number of category values for a categorical attribute. The cardinality values for all categorical attributes are listed in Table 18.

Table 18: Cardinality of categorical data attributes

Attribute	Number of categories
BU	7
Weekday	7
Item	1092
Material/part number	364964
Material group	663
Plant	86
Vendor	6698
Activity	13

It shows that the cardinality is extremely high for attributes item, material group, material/part number, plant, and vendor. The high cardinality likely leads to problems when encoding these attributes. For example, the dimension of the feature vector would explode if one-hot encoding is used, since an extra feature would be created for each of the category values.
To further asses the proportions of the category values within the dataset, the distributions have been plotted. For the attributes with a cardinality value higher than the value of 'plant' (86), only the 86 most frequent categories are displayed. These distributions are shown in Figures 37 - 44 in Appendix A. Note that due to confidentiality the category values have been anonymized for BU, vendor, plant, material/part number, and material group. It is noticed that for some attributes the values are not that spread out over the categories. For example, 2 out of the 1091 item categories (Item 00010 and 00001) already account for almost 50% of the data. Moreover, 3 out of the 7 categories of BU account for 90% of all values. Besides, clearly one plant is where most PO items need to be delivered, having a peak in the plot at around 21%. Another interesting observation is that the least activities in the P2P process are performed during the weekend (value 6 and 0), however, still a small amount of activities are executed.

The preprocessing steps that are performed for the potential input features based on the observations within this univariate analysis, are explained in Section 4.3.3. But, first the bivariate analysis is described within the subsequent section investigating the relations among the potential input features and with the target variable.

4.3.2 Predictive Part - Bivariate analysis

The goal of the bivariate analysis is to understand the relation between the features and the target variable. First, the relation between the categorical features and the target variable is analyzed, followed by the relation between the numerical features and the target variable. Different methods were used for the two types of relations. The results are explained in the next paragraphs.

Relation Categorical Features & Numerical Target
The ANOVA was used to get an indication of the relation between the categorical features and the target variable. In case of the cardinality for a feature being higher than 20, only the most frequent 20 categories were included to reduce computational time. For each categorical feature an F-score and p-value was calculated, as explained in 4.1. The results for all the categorical features are summarized in Table 19.

Table 19: ANOVA results categorical features & target variable

Attribute	F-score	p-value
BU	60491.301	0
Weekday	78.991	3.460e-99
Item	6584.29	0
Material group	9632.471	0
Material/part number	9903.965	0
Plant	22150.811	0
Vendor	12999.122	0
Activity	4029.789	0

The ANOVA suggests a relation with the target variable for all categorical features, since a significant difference in mean target value is found between at least two of the categories for the features (p-value ≤ 0.05) (Glen, 2022a). The highest F-scores are noticed for features BU, plant, vendor, and material number/part number, each having a score above 10,000. These categorical features are expected to have the strongest relationship with the target variable. The feature weekday stands out in a negative way, having the lowest F-score of 78.991.

Relation Numerical Features & Numerical Target
The relation between the numerical features and target variable was examined using two different methods: (1) a correlation matrix and (2) a scatterplot. The correlation matrix shows the extent to which there is a linear relation between two numerical features (Muthukrishnan, 2021). This matrix is shown in Figure 45 in Appendix A. It depicts the correlation between each pair of numerical features and between each of the numerical features and target variable. The cross correlation between numerical features is examined to see whether certain numerical features are highly correlated indicating that one of the two features might be redundant. The highest correlations are found between time since last event and time since case start (0.640), as well as between OTIF of last year and OTIF of last month (0.420). Both values are positive, meaning an increase for one factor leads to an increase in the other factor, and the other way around (Muthukrishnan, 2021). Furthermore, it is found that none of the features seem to have a linear dependency with the target value. This is visible in Figure 45, where the shown correlation values of the numerical features with the target variable are extremely low. The scatterplots in Figures 46 - 61 in Appendix A confirm this finding. For some features it looked like there was a linear relation, such as for the difference in days between PO creation and contractual delivery date. The scatterplot gives the impression that higher values for this feature are related to lower values for the target. However, when looking at the target variable in its normal range, so without the outliers shown in Table 17, no linear relation can be detected anymore. Possibly, the numerical features alone have almost no influence on the target variable, however, when combined an effect might arise.

The preprocessing steps that are performed for the potential input features based on the observations within the univariate as well as bivariate analysis, are explained in the next section.

4.3.3 Predictive Part - Resulting Preprocessing Steps

Since material number has around 24% of missing values, the attribute is chosen not to be considered as input feature. Material group and a combination of material number, part number, and short text have been investigated as possible substitutes for material number. It has been observed that material/part number has only 8 missing values compared to 254 for material group. However, the cardinality is much higher which might raise issues during the feature encoding. But, within the bivariate analysis a much stronger relationship with the target variable is indicated for material/part number compared to material group. Therefore, material/part number is kept as possible feature and material group is dropped. Furthermore, it is decided to remove the OTIF of last month from the feature set, due to containing a large number of missing values, as well as having a high correlation with OTIF of last year. Lastly, the missing values are removed for the remaining attributes with missing values, material/part number and OTIF of last year. By removing these missing values for material/part number and OTIF of last year, 4.03% of the dataset is deleted. However, this should not be a problem due to the large dataset containing 9,765,979 P2P prefixes.

Moreover, the detected outliers in Table 17 are removed from the dataset. For the attribute 'number of days between PO creation and contractual delivery date' this also means that the negative values are removed. The attribute 'number of changes to delivery date' has been inspected in more detail, since removing all values higher than zero makes the attribute redundant considering it will only have one value and does not give information about changes made to the delivery date anymore. Therefore, the frequency is checked for each of its possible values. It is evident that the frequencies are highest for value 0, 1 and 2, with a frequency of respectively 9317437, 439539, and 6827. These values account for 99.999% of all data points for this attribute. Hence, to keep some information about the number of changes, only the very infrequent values are removed, which are the values higher than 2. In total 13.351% of the dataset is eliminated with the outlier removal. Since the event log contains over 9 Million prefixes, still a high volume of data is left.

Finally, after the removal of all missing values and outliers a total of 1,187,773 cases are present in the dataset and 8,157,240 prefixes.

To decide whether a data transformation is necessary for the numerical features the skewness values are examined after removing the missing values and outliers.

Table 20: Skewness of the numerical attributes after removal of missing values and outliers

Attribute	Skewness
OTIF of last year	-0.617
Number of PO lines per vendor	0.896
Number of changes to delivery date	4.295
Time since last event	4.442
Time since case start	3.276
Time since midnight	0.016
Difference in days between PO creation and contractual delivery date	1.697
Target variable	0.035

Clearly, no extreme values like in Table 15 are present for skewness anymore. The skewness values decreased most for attributes time since last event and time since case start, with a difference of respectively 122.987 and 53.024. Nevertheless, still 4 out of the 8 attributes have a skewness outside the range of [-1,1] implying a very assymmetrical probability distribution for these attributes (Gawali, 2021). Consequently, log-scaling is performed for all numerical attributes, except the target variable. Although some of the other attributes had a low skewness value, still a log-transformation was applied to all numerical input features in order to be consistent. The target values are initially not log-transformed since the skewness value is low. However, for all predictive models one experiment has been performed with log-scaling the target variable as well, as will be later explained in Section 5.1.1.

Lastly, the bivariate analysis indicated a possible relationship between feature and target only for the categorical features and not for the numerical features. Especially, BU, plant, vendor and material number/part number seemed promising features. However, the ANOVA and correlational analysis both analyzed the relation between a feature and target variable in isolation, meaning no interaction effects between features have been explored. Therefore, this analysis is not enough to create the final selection of features to use as input for the predictive models. Hence, automatic feature selection methods are used to further explore the impact of the possible input features on the target variable when combining the features. This additional feature selection step is elaborated in the next section.

4.3.4 Predictive Part - Feature selection

As explained in Section 4.1, four different automatic feature selection methods have been tested in order to explore which features are likely to have a high impact on the target variable when used as input for the predictive models. The input data used for these predictors has been preprocessed using the steps mentioned in Section 4.3.3. The first method is a Random Forest Regressor and selected 12 features as being most impactful, like shown in Table 21. For the other methods, The Ridge Regressor with cross-validation, Lasso Regressor with cross-validation, and sequential feature selector, the feature threshold was set to 30 and therefore the 30 most important features were selected. All selected features by the different methods are summarized in Table 21. The features given in Table 21 are not ranked, but listed in arbitrary order.

Table 21: Feature selection results

Random Forest Regressor: importance	Ridge regression with cross-validation: importance	Lasso regression with cross-validation: importance	Ridge regression with cross-validation: forward selection
OTIF of last year	OTIF of last year	OTIF of last year	OTIF of last year
PO lines per vendor	Difference in days between PO creation and contractual delivery date	Difference in days between PO creation and contractual delivery date	PO lines per vendor
Difference in days between PO creation and contractual delivery date	BU - 3 categories	BU- 3 categories	Difference in days between PO creation and contractual delivery date
time since last event	plant - 5 categories	plant- 6 categories	time since last event
time since case start	material/part number - 9 categories	material/part number -7 categories	time since case start
time since midnight	vendor - 4 categories	vendor-5 categories	time since midnight
weekday 1-5	item- 1 category	item- 1 category	BU - 1 category
activity- 1 category	activity-6 categories	activity-6 categories	plant-4 categories
			material/part number -5 categories
			vendor -3 categories
			item -3 categories
			activity -7 categories

Table 22 summarizes how often each of the features is selected by the various feature selection methods.

Table 22: Feature counts

Feature	Number of times feature is selected
OTIF of last year	4
Number of days between PO creation and contractual delivery date	4
Activity	4 (max. 7 out of 13 categories selected)
BU	3 (max. 3 out of 7 categories selected)
Plant	3 (max. 6 out of 9 categories selected)
Material/part number	3 (max. 9 out of 11 categories selected)
Vendor	3 (max. 5 out of 9 categories selected)
Item	3 (max. 3 out of 8 categories selected)
PO lines per vendor	2
Time since last event	2
Time since case start	2
Time since midnight	2
Weekday	1 (max. 5 out of 7 categories selected)
Number of changes done to the delivery date	0

The most impactful features according to the automatic feature selection methods are the following: OTIF of last year, PO lines per vendor, number of days between PO creation and delivery date, time since last event, time since case start, time since midnight, BU, plant, and material/part number. Therefore, these features are referred to as the top features. Both OTIF of last year and the number of days between PO creation and delivery date are selected by all four methods. Furthermore, BU, plant, and material/part number are selected by three of the four methods and for a high percentage of categories. Additionally, the PO lines per vendor and time features are selected by two of the four methods.
Nonetheless, different results are retrieved for the four methods and the selected features are just an indication of possible influential features. Therefore, the experiments for the predictive models are mainly executed with all the features as listed in Table 7, except for the earlier removed OTIF of last month and with the substitute for material number, i.e. material/part number. Additionally, for each of the predictive models an experiment is performed with only the top features, as listed above.

4.3.5 Prescriptive Part - Univariate Analysis

Like explained in Section 4.1, only the univariate analysis was performed for the prescriptive part of this thesis. This section explains the results of the univariate analysis, focusing on the descriptive statistics of the data attributes of interest, missing values, as well as outlier values. This analysis is mainly performed to get an indication of the data quality for the possible input features and to investigate the effect of quality issues on the existing expedite framework of Company X.

The data for the possible input features is retrieved by selecting all PO items that were not yet delivered, not deleted, and were goods. This resulted in a selection of 164,473 PO items. The next paragraph discusses the missing values for the various features.

Missing Values
Missing values are found for the attributes material criticality, stock on hand, OTIF of last month, as well as OTIF of last year. The exact number of PO items for which there are missing values of an attribute and corresponding percentages are summarized in Table 23

Table 23: Missing values for the various prescriptive attributes

Attribute	Number of cases with missing value	Percentage of cases with missing value
Material criticality	122675	74.587
Stock on hand	132561	80.597
OTIF last year	3966	2.411
OTIF last month	38868	23.631

Especially high percentages of missing values are found for attributes material criticality and stock on hand. According to the process mining expert it means that the master data for these attributes is not well filled. However, the weighted scoring model to calculate the expedite score accounts for missing values for these attributes. In case the material criticality has no value, the material criticality score in the weighted scoring model gets a value of 2. This lowers the priority for expediting a PO item. If stock on hand has a missing value, the stock vs non-stock score in the weighted scoring model will get a value 5. It was expected that stock on hand values would only be missing for non-stock items, since these items are not supposed to have stock. However, missing values are also found for stock items. It could then happen that items are getting a stock vs non-stock score of 5 while it was actually supposed to get a score of 10, since there was no stock.
Moreover, the OTIF of last year has a lower percentage of missing values than the OTIF of last month. According to the process mining expert, the supplier historical delivery performance in the weighted scoring model gets a value of 10 in case the OTIF last year has a value, but the OTIF last month value is missing. Besides, when the OTIF last month is available but the OTIF last year is missing, a score of 0 is assigned to the supplier historical delivery performance. However, both of these cases can create a real bias in the expedite framework. For example, when the OTIF of last

month is missing and the OTIF of last year is available, the supplier historical delivery performance factor gets a value of 10, which increases the expedite score with 2 points, which in turn increases the priority for expediting enormously while it is not even justified.

Descriptive Statistics
Next, the descriptive statistics are provided in Table 24 for the numerical attributes.

Table 24: Descriptive statistics numerical attributes

Attribute	Mean	Median	Mode	Min	Max	Standard Deviation	Skewness	Kurtosis
OTIF of last year (%)	46.274	49.728	49.728	0	100	22.114	-0.075	-0.339
OTIF of last month (%)	71.399	77.114	77.114	0	100	27.608	-1.292	0.940
Stock on hand	8088.490	0	0	0	1.722e+06	93021.7	14.447	220.366
Days overdue score	5.177	5	10	0	10	4.755	-0.051	-1.920

For both the OTIF of last year and OTIF of last month it seems unlikely that there are outlier values. The mean and median values are very similar and the kurtosis is not extreme (Gawali, 2021). The same holds for the days overdue score. This is confirmed by using the z-score, which checks whether there are outlier values that are more than 2 standard deviations away from the mean.
However, the stock on hand definitely seems to contain outlier values. The mean value is much higher than the median. When looking at the maximum value, 1.722e+06, it seems very likely that this is the cause for the high mean value. Furthermore, the values for skewness and kurtosis are extremely high. However, when using the z-score, no outliers are found for stock on hand. The underlying reason is the fact that the z-score labels a value as an outlier when the value is more than 2 standard deviations away from the mean. In the case of stock on hand, both the mean and standard deviation have a very high value. This means that even the maximum found value is not more than 2 standard deviations away from the mean. Fortunately, these extreme values for the stock on hand do not cause problems in the calculation of the expedite score. An item only receives a high priority for expediting based on the stock vs non-stock factor when the item is a stock item and if the stock on hand is 0. So, when the available stock is extremely high, it does not result in a higher stock vs non-stock score compared to when the stock would be for example 1.

Furthermore, Table 25 shows the cardinality of the categorical attributes and the distribution of the different category values.

Table 25: Cardinality of prescriptive categorical attributes

Attribute	Cardinality	Category values	Percentage
Stock vs non-stock	2	Stock	16.692
		Non-Stock	83.309
Material criticality	9	1	3.895
		C1	3.993
		2	4.957
		C2	21.276
		3	6.797
		C3	42.284
		4	2.670
		5	4.861
		NC	9.266

The distribution of the values for stock vs non-stock is clearly unbalanced, where the non-stock items are the majority of values with 83.309%. This means that the stock versus non-stock score in the weighted model for expediting has at least for 83.309% of the PO items a score of 5 (see Section 3.4 for the weighted scoring model). Material criticality has a higher cardinality, since it contains 9 unique category values. The most frequent values are C3 with 42.3% and C2 which accounts for 21.276%. This means that the material criticality score in the weighted scoring model is most frequently has a value of 4, followed by a value of 6.

Invalid Values
An interesting observation is that non-stock items still can have a stock on hand value. However, this should not be possible according to the process mining expert. On the other hand, it does not lead to an issue within the calculation of the expedite score, since the non-stock items would already get the lowest score for the factor stock vs non-stock in the weighted scoring model, and the stock on hand does not further influence this value. Moreover, it was observed that there were PO items in the selection for which the planned delivery date was due a very long time ago, in the year 2013 and 2016. Therefore, the days overdue score within the weighted factor model gets the maximum value. These PO items need to be removed from the analysis when evaluating the prescriptive model later in the project.

Overall, since no major issues arose during the univariate analysis concerning the possible input features, all features are selected to be used within the prescriptive model.

5 Predictive Process Monitoring

This chapter elaborates on all the steps performed for the predictive part of this thesis, as summarized in the research methodology framework in Figure 3. First the methodology behind the performed steps is explained. These steps are data preparation, model selection and implementation, evaluation and deployment (Becker et al., 2020). Data preparation consists of multiple sub-steps which are input feature encoding, event log sampling, prefix generation, splitting the data into a train, validation, and test set, as well as over/undersampling. Moreover, the model selection and implementation step is explained for each selected predictive model seperately. Subsequently, the results are provided for two data preparation steps, the event log sampling and data splitting. Next, the results are provided for the experiments and analysis of all predictive models, as well as for the deployment of the best model.

5.1 Predictive Process Monitoring Steps

This section explains the method used for the steps performed regarding the predictive part of this thesis. These steps are data preparation, model selection and implementation, evaluation and deployment.

5.1.1 Data Preparation

The next step after business and data understanding for the predictive process monitoring part of this thesis is data preparation. According to Becker et al. (2020), the first task in this phase is to select the required data. This was accomplished by applying the earlier defined process filters for model training (see Table 9) and by retrieving the selected features as described in Section 4.3.4. Next, the problems regarding data quality were addressed by cleaning the data. Outliers and missing values for attributes were removed based on the findings of the EDA (see Section 4.3.3). Thereafter, the data for all attributes was further preprocessed by performing the following steps: a) encoding of the categorical features as well as the numerical attributes, b) creating a subsample of the entire dataset, c) transforming the cases into a prefix format, d) splitting the data into a train, validation and test set, and e) optionally resampling of the data. The various steps are explained in more detail in the upcoming paragraphs. The last step during data preparation was to export the prepared dataset to the format suitable for the method selected in the next step (Becker et al., 2020). Since Python was used to develop the predictive models, the input format was chosen to be a CSV-file. This file format can easily be imported like a data frame into Python using the Pandas library (NumFOCUS, 2022a).

Predictive process monitoring is applied in this master thesis, meaning that predictions are made for what is going to happen in the future with an ongoing case, given the prefix of this case (Rama-Maneiro et al., 2021a). In our case it means that the number of days between the actual delivery date for PO items and the contractual delivery date was predicted using the event prefixes for the different items (remember that the concept of a prefix has been explained in Section 2.1.1). Splitting the cases into prefixes has as advantage that the predictive models can later be analyzed on how well they perform for different prefix lengths, thus for different time horizons, and for different information availability. It also helps the predictive models to take into account the temporal order of events in the event log. The exact procedure of creating prefix feature vectors is explained in detail in the "Prefix Generation" paragraph. Before these prefix feature vectors can be created, the input features need to be encoded and a subsample needs to be created from the total dataset.

Encoding of Input Features
As mentioned in the literature review (van Wijlen, 2022), inputs need to be modified in order to be able to use them in a neural network. The reasoning behind this is the fact that the inner logic of neurons is designed for input vectors with values between -1 and 1 and a mean of 0. The process of transforming the input data to this range is called data encoding. An important distinction between encoding methods is whether the attribute being encoded is categorical or continuous.

Encoding is most important for categorical attributes since a numerical representation is needed in order to give it as input to the neural network. However, even if the attribute is already numerical as for the continuous attributes, the value is often transformed to a better suiting format. First, the method for encoding the numerical attributes is discussed, followed by the two methods used for the categorical attributes.

Normalization
By normalizing numerical data, the possible values for a variable are scaled down to fall in smaller range. The goal is to change the values of numerical attributes to use a common scale without altering differences in the ranges of values or losing information (Karra, 2022). According to Sola and Sevilla (1997) adequate normalization of input prior to the training process is critical to generate good results and reduce computational time. This is especially necessary when numerical attributes have a highly varying range of possible values. Otherwise the DL algorithm will be dominated by the features that use a larger scale, negatively affecting model performance. Since this is the case for most of the numerical attributes (see Section 4.3.1), normalization was performed for all numerical input attributes within all experiments conducted with the predictive models. Besides, one experiment was added for the predictive models in which the target values were normalized as well. The chosen technique for normalization was log-scaling. This means that the log is computed for the feature values, see Equations 3 and 4.

$$x' = \begin{cases} log(x), & \text{if } x>0 \quad (3) \\ log(x+1), & \text{if } x\geq 0 \quad (4) \end{cases}$$

Log-scaling is helpful in our case since the data for the numerical attributes is skewed (see Table 20). This implies that a small amount of values have many points, while most other values have few points. Log-scaling makes the distribution of the data less skewed and reduces variability (Lindgren, 2019).

Each of the categorical attributes must be encoded into fixed feature vectors individually representing them. It was seen in van Wijlen (2022) that one-hot encoding is most popular, followed by embedding. Therefore, these feature encoding methods were used in the experiments for the predictive models. Both methods are explained in more detail in the upcoming paragraphs.

One-hot encoding
Due to one-hot encoding being so widely used this was the first and main method used to encode the categorical attributes. Before executing the one-hot encoding method, another preprocessing step was required. The categorical attribute values were anonymized to ensure no confidential information, e.g. vendor or business unit names, is being leaked when using a cloud-based server to run the experiments on in the end. The anonymization process was performed in Python by first creating a dictionary containing for each categorical attribute a dictionary with all possible category values as keys, and string labels in range [1, number of unique category values] as items. Subsequently, for each categorical attribute the dictionary was applied to the values for the categorical attribute to give all values the corresponding label.

Next, the one-hot encoding was performed, where each categorical attribute was represented in a binary vector, where the size of the vector was equal to the total number of possible distinct values for that attribute (Rama-Maneiro et al., 2021a). An additional feature was thus created for each unique category value for the attribute. When the number of unique category values, i.e. cardinality, for an attribute was extremely high (see Table 18), additional features were only created for a percentage of the unique category values, where the other categories were grouped in the category 'other'. This percentage was initially set to 5%. However, this still gave a high number of categories for some attributes. Therefore, the percentage was lowered to 3%, 2% and 1.5% for other attributes. Otherwise, the number of features would explode, by all the extra features created for the in total 7 selected categorical attributes (see Section 4.3.4). Subsequently, for each event such a binary feature vector was created. For this, the *get_dummies* function from the Pandas library was used in

Python (NumFOCUS, 2022c). The position in the vector took value one if the considered attribute value for the event corresponded to that position. The other positions were then filled with zeros (Harane and Rathi, 2020). An example for the attribute 'Business Unit' is shown in Table 26. This category contains seven different values where the values were anonymized by converting them to string values ranging in $[1, 7]$. Hence, the feature vector length was seven for each of the events in the prefix (or entire trace).

Table 26: One-hot encoding example of attribute 'Business Unit'

Original trace	After one-hot encoding
$<$ '1','2','4','5'$>$	$<< 1,0,0,0,0,0,0 >, < 0,1,0,0,0,0,0 >, < 0,0,0,1,0,0,0 >, < 0,0,0,0,1,0,0 >>$

A disadvantage of one-hot encoding is that it gives high sparsity when there are categorical attributes in the dataset with a high number of unique category values (Kumar, 2022). Since this is the case for several categorical attributes (see Table 18 in Section 4.3.1), embedding encoding was tested as well. This method is well-known for its advantage of dimensionality reduction and less memory usage (Rama-Maneiro et al., 2021a).

Embedding
Embedding is a method in which event traces are treated similar to natural language sentences and events like words. With the embedding encoding a matrix with $v \times m$ dimensions was created, where v stands for the size of the vocabulary (all possible attribute values) and m for the embedding space (Harane and Rathi, 2020). Each row in the embedding matrix corresponds to a category value of the attribute and is actually an embedding vector with a fixed size. This vector's dimensionality is represented by the number of columns in the embedding matrix and refers to the number of neurons in the neural network's hidden layer (Rama-Maneiro et al., 2021a; Weinzierl et al., 2020b). It was chosen to learn the parameters of the embedding matrices simultaneously with the NNs in the method implementation phase. This was accomplished by adding a seperate embedding layer for each of the categorical attributes in Python either with the Tensorflow or Pytorch library (Tensorflow, 2021; PyTorch Contributors, 2022a). This depended on the library in which the predictive model was built. More details about the embedding layers, their chosen parameters, and integration in the NNs are provided in Section 5.1.2. After training these embedding matrices in parallel with the NNs, the vectors in these matrices show the similarity of attribute values (Weinzierl et al., 2020b).
Before the embedding matrices for the categorical attributes could be learned through the embedding layers in the networks, the feature vectors including these attributes needed to be transformed into prefix format, which could be provided as input to these embedding layers. However, this part is discussed in the second next paragraph ("Prefix Generation"). In order to build the feature vectors for the categorical attributes, the values for each of the attributes had to be labeled with an integer value in the range [1, number of unique category values]. This preprocessing step was seen in multiple PBPM approaches applying embedding encoding (Arat, 2019). The labeling was performed in Python by applying for each categorical attribute a dictionary mapping the unique category values to an integer labels in range [1, number of unique category values]. Finally, each feature vector for an attribute is then the vector of the integer labels for all of the events in the trace.

Event log sampling
In order to reduce computational time for training the predictive models, it was decided to create a subsample from the entire dataset. Like Fani Sani et al. (2021) mention in their paper, reducing model training time is especially relevant for predictive business process monitoring techniques, since efficiency and reliability of results need to be tested through repeated training of distinct prediction models with different parameters. The method used for selecting instances in the subsample is designed based upon the method of Fani Sani et al. (2021). The goal of instance selection is to

reduce the original dataset to a workable size to perform ML tasks, while maintaining similar performance as when the original dataset is used. The schematic view of the event log sampling procedure is depicted in Figure 9. The entire procedure was executed in Python. In order to have a representative subsample of the entire dataset, it is important to keep the distribution of the features and target variable similar to the entire dataset when creating a subsample. Therefore, the main idea of the event log sampling procedure is to group the cases into case variants, where the case variants are based upon the control-flow as well as input feature values and the target values. Subsequently, a divisor k is chosen where for each variant a fraction of one k^{th} of all cases belonging to the case variant is randomly selected. For example, when using a divisor $k = 3$ and having case variant 1 containing 100 cases and case variant 2 containing 300 cases, there are $\lceil \frac{1}{3} * 100 \rceil = 34$ cases randomly selected from variant 1 and $\lceil \frac{1}{3} * 300 \rceil = 100$ cases from variant 2. To find the right balance between the decrease in computational time and loss of performance, multiple subsamples of different sizes were created by changing the value of k. These predictive models were trained on the different subsamples to see how performance was affected by the subsample size.

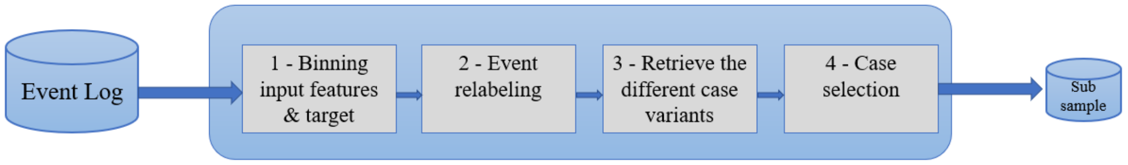

Event Log
1 - Binning input features & target
2 - Event relabeling
3 - Retrieve the different case variants
4 - Case selection
Sub sample

Figure 9: Event log sampling procedure

In the following, each of the steps is elaborated in more detail.

1. *Binning input features and target*: In the first step, all values for the input features and for the target variable have been split up into bins in Python. The bins created for the categorical variables were similar to the dummies created in the one-hot encoding step. For the numerical attributes, the bins were created using the *qcut*-function from the Pandas library in Python (NumFOCUS, 2022d). This automatically divides up the values for a variable into q bins with an equal percentage of data in it. The event attributes, except for 'Activity', were excluded from this procedure. Since, these attributes have a different value for each event, it would lead to a lot of unique case variants in step 3. This is undesired as it would increase the smallest possible sample size.

2. *Event relabeling*: In the second step a new feature was created which replaces 'Activity' as usual feature resembling 'concept:name' in the event log. This new feature takes on a value for each event in the event log and is a combination of the bin values for all attributes and target for the specific event. For example, for one of the events the attribute 'concept:name' was now changed from 'Create Purchase Requisition' to 'Create Purchase Requisition Item-bin_nrchanges1bin_ttmotif1bin_polines1bin_delta_po_dd1bin_Days too late1bu_1plant_others vendor_others item_1MatnrShort_others'. By using this 'concept:name' in the event log instead of the 'concept:name' containing only the 'Activity', the case variants were determined based upon not only the activity feature, but on all feature values as well as the target. These variants then sort of cluster the traces based on feature and target values.

3. *Retrieve the different case variants*: In this third step, the unique variants of the event log and the corresponding traces were detected. These unique variants were found using the PM4PY-library in Python (Fraunhofer FIT, 2021b). First, the dataframe containing the event log was reduced to only containing the typical event log attributes: case ID, event time, and concept: name, where concept:name was the new feature as created in the second step. This dataframe was converted into a log object in Python. Subsequently, the unique variants were retrieved from the log object by using the *get_variants_as_tuples* function from PM4PY.

4. *Case selection*: In the final step, cases were selected from the different variants. For each variant a fraction of one k^{th} of all cases belonging to the case variant was randomly selected. The pseudocode for how the cases are selected using Python is shown in Algorithm 1.

Algorithm 1: Pseudo-code of case selection step in the event log sampling procedure

```
1  Initialize $k$ ;                                                   ▷ $k \in (3, 6, 12, 24)$
2  selected_cases=[] ;                                 ▷ Create list for the selected cases
3  for variant in range of the number of variants do
4      traces= all traces belonging to the case variant;
5      nrcases=$\lceil len(traces)/k \rceil$;
6      randomlist=[] ;                ▷ Create list for the random selected indices for traces
7      try:
8          randomlist+=random.sample(range(len(traces)),nrcases) ; ▷ Add randomly selected trace indices for
            the number of cases specified by variable nrcases
9      except:
10         randomlist=[i for i in range(len(traces))] ;   ▷ Keep all traces if variant contains exactly as many
            traces as variable nrcases
11     end
12     variantcases=[traces[idx].attributes['caseID'] for idx in randomlist];     ▷ Collect case ID for the selected
        traces
13     selected_cases+=variantcases ;     ▷ Add selected cases for the variant to the list of all selected
        cases
14 end
15 subsample=dataframe[dataframe['caseID'].isin(selected_cases) ;  ▷ Obtain subsample of event log by keeping
    only the selected case IDs
```

Prefix generation

In this data preparation step, the cases in the subsamples and total dataset are split into event prefixes (see Section 2.1.1 for the concept of a prefix). Furthermore, these event prefixes are converted into tensors which the neural networks can handle as input. This process is called sequence encoding (Rama-Maneiro et al., 2021a). The two sequence encoding methods that were used among the predictive models, prefix padding and aggregation encoding, are explained in the next paragraphs.

Prefix Padding

The most used encoding method is prefix padding (Rama-Maneiro et al., 2021a). Here, tensors of a fixed vector length are created for each event prefix. The event prefixes are padded with zeroes in case they are shorter than the specified vector length. The vector length is often set to the length of the longest trace in the log. It was found in literature that different DL models use slightly different versions of prefix padding. For example, different shapes of vectors are seen for the prefixes, like (1,max_tracelength), (n_features, max_tracelength), and (max_tracelength,n_features). An example of a more frequently observed type of prefix padding encoding is shown in Table 27. It presents the encoding for prefixes of different lengths belonging to one case, where maximum trace length is 4. The columns represent different features in the vector (features from the selected main feature set as given in Section 4.3.4) and v_{exy} stands for value (v) of event (e) number x for feature y. When the prefix length increases, the feature values for the new event are added filled in at the bottom row of the feature vector and the feature values for the previous events shift one row up.

Table 27: Prefix padding example

Prefix length	f1	f2	f3	f4	f5
1	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	v_{e11}	v_{e12}	v_{e13}	v_{e14}	v_{e15}
2	0	0	0	0	0
	0	0	0	0	0
	v_{e11}	v_{e12}	v_{e13}	v_{e14}	v_{e15}
	v_{e21}	v_{e22}	v_{e23}	v_{e24}	v_{e25}

In the case of embedding, the vectors look slightly different than in Table 27. Instead of using a single feature vector like in Table 27 containing all the features for a prefix, a separate feature vector was created including all numerical features and n feature vectors were constructed for the n categorical features. First, one prefix padded feature vector was built for all numerical features of shape (n_features, max_tracelength). Secondly, separate prefix padded feature vectors were created for each of the categorical features having a shape of (1,max_tracelength). An example of prefix padded feature vectors for the prefixes of one case and for categorical attribute 'Business Unit' is shown in Table 28. Its maximum trace length is 4. These prefix feature vectors are then given as input to the predictive models.

Table 28: Prefix feature vectors - embedding

Original trace	Prefix length	Prefix padded vector
< '1','2','4','5'>	1	$< 0, 0, 0, 1 >$
	2	$< 0, 0, 1, 2 >$
	3	$< 0, 1, 2, 4 >$
	4	$< 1, 2, 4, 5 >$

Aggregation encoding
For several predictive models it was necessary to have an input of shape (n_prefixes, n_features), meaning a 1D-representation of each prefix. An appropriate sequence encoding method was chosen based on the paper of Teinemaa et al. (2019). The paper reviewed the performance of predictive process monitoring methods for different types of sequence encoding methods which were in line with the required input shape for the predictive models. The aggregation encoding method was selected to generate this prefix representation, since it exploits information from all performed events (Teinemaa et al., 2019). A disadvantage is that order of events is neglected. However, the dimensionality does not explode substantially, which is for example a problem with index encoding where the order is taken into account. Moreover, the best predictive performance is usually achieved using the aggregation encoding (Teinemaa et al., 2019).

The main idea of aggregation encoding is that aggregation functions are applied to the values for the event attributes, whereas values for the case attributes remained in their original shape. Possible aggregation functions for numerical features are min, max, mean, sum, and standard deviation, whereas for categorical features these are frequencies or occurrences (Teinemaa et al., 2019).
Since the frequencies aggregation method is superior compared to the boolean occurrences method (Teinemaa et al., 2019), the one-hot encoded feature representation of the categorical event attributes 'Weekday' and 'Activity' is transformed into a frequency representation. An example for 'Weekday' is shown in Table 29.

Table 29: Aggregation encoding example

Version	Case	Event	Weekday 0	Weekday 1	Weekday 2	Weekday 3	Weekday 4	Weekday 5	Weekday 6
		e1	0	1	0	0	0	0	0
Original	ID1	e2	1	0	0	0	0	0	0
		e3	0	1	0	0	0	0	0
After		e1	0	1	0	0	0	0	0
frequency	ID1	e2	1	1	0	0	0	0	0
aggregation		e3	1	2	0	0	0	0	0

The numerical event attributes (time since last event, time since case start, and time since midnight) were firstly not transformed with an aggregation function. The reason is that the numerical attributes are already rather aggregate measures since information of previous events is taken into account in the value. For example, time since case start considers the time information of all events since the beginning of the case. Since, the aggregate character was less obvious for the features time since last event and time since midnight, it was as well tested to use mean time since last event and mean time since midnight as input features. The results are discussed in Section 5.2.3.

Train-Validation-Test Split

After the subsamples were created to be used in the experiments and the feature vectors in these subsamples were transformed into a prefix format, the data in each of the subsamples as well as the total dataset could be split into a train, validation and test partition. When building and implementing machine learning models it is important to split the used dataset into different datasets, so the models can be evaluated on data that was not used during the training phase. Otherwise problems might arise like overfitting without noticing it. Therefore, within each experiment, the used dataset (the prefix dataset X and target vector y) was split into a train dataset, validation dataset, and test dataset. The train dataset was used to train the predictive model, which are for example the weights and biases in the case of a NN. The models sees this data and learns from it. Next, the validation dataset was the sample of data used to give an unbiased evaluation of the model fit on the training dataset, for a specific model being trained. The validation data was used to evaluate whether the model was overfitting or not and to optimize the model in terms of hyper-parameters. It was then mainly used to frequently evaluate a model that was being built and thus to help during the "development" stage of the model (Tarang, 2017). Lastly, the test dataset was the sample of data used to evaluate a **final** model fit on the training dataset, for the different predictive models (Tarang, 2017). The test dataset was only used after the model training was finished. The different competing predictive models could then be evaluated using the test dataset. The results on the test dataset decided which of the models performed best.

Already before the encoding of the input features, the data was sorted on the event time attribute. This sorting was performed to ensure that the data was in chronological order, and that past data was selected for training and future data for testing. Next, the dataset (consisting of the prefixes and the target values) was split into a train dataset and test dataset, where 67% was retained for training and 33% for testing. The split was performed by using the *train_test_split* function from the Sklearn Python library (scikit-learn developers, 2022e). No shuffling was applied in order to keep the temporal order and select the first 67% of the dataset for training and the last 33% of values in the dataset for testing. Subsequently, the train dataset was split again where the first 80% is kept for training and the other 20% for validation. Again, no shuffling was applied to remain the temporal order of the data. Since the split was made on a prefix-level, it could happen that prefixes from a case in the train set could be in the train set as well as in the validation set. The same holds for the prefixes from a case in the validation set, where they could be in both the validation and test set. However, this could only happen for prefixes from the last case within the train or validation set due to the following. Within all experiments, the prefixes in the used dataset were not only chronologically sorted but also grouped for each case. Therefore, the prefixes belonging to the same case are together in the dataset and at the moment of splitting it could happen that the split is made not exactly before the first prefix of a case or after the last prefix of a case. Table 30 shows an example of chronologically ordered and grouped prefixes like in the used datasets and what can happen when the split is performed on prefix-level.

Table 30: Explanation of possible data leakage while splitting

Case	Prefix	Partition of used dataset
ID1	p_length1	
ID1	p_length2	Train
ID1	p_length3	
ID1	p_length4	
ID1	p_length5	
ID2	p_length1	Valid
ID2	p_length2	
ID2	p_length3	

To prevent data leakage between the different sets due to these prefixes, a check was performed on the validation dataset and test dataset. For example, it was examined whether the first prefixes in the validation dataset were containing information only for the first event. This would indicate that the dataset starts with a new case. If not, the examination was iterated for the next prefix in the set until the first prefix of the new case was found. The indices for the "not first" prefixes were stored and could then be used to remove the prefix values and target values from the validation set and add them to the train set. The same procedure was performed for the test dataset.
After the split was completed, the distribution of the target values was investigated for the train, validation, and test dataset. This was done by plotting a histogram of the target values for the three datasets with the Matplotlib-library in Python (Matplotlib development team, 2022). The histograms showed the distribution of the existing target values in the dataset in percentages. The distributions for the datasets were compared to each other, as well as compared to the distribution of the target values before splitting the dataset. It is an important practice, since first of all the model needs to be trained and evaluated on data which is similarly distributed as the non-split dataset. Moreover, the model will not be able to make good predictions on the test set if the target values are dispersed differently in the train set than in the test set.

Oversampling/Undersampling
Oversampling and undersampling are useful methods when the data is imbalanced. Imbalanced data implies that there is an unequal distribution of data instances over classes in the dataset (Mohammed et al., 2020). Over- and undersampling are resampling methods that add records to the minority class or delete instances from the majority class (Mohammed et al., 2020). For example, oversampling techniques attempt to multiply examples or generate new examples in the dataset such that effects of discrimination are diminished (Rančić et al., 2021).
However, since our predictive task is of the regression type rather than classification, regression over imbalanced data means predicting rare and extreme continuous target values from input data (Neptune Labs, 2022). Nevertheless, regression over imbalanced data is not that researched in literature as classification (Torgo et al., 2013). Therefore, an own interpretation of random over- and undersampling was implemented. First, the target variable was assigned a class by automatically splitting up the target values into eight equally distributed classes using the *q_cut*-function from Pandas (NumFOCUS, 2022d). Next, the prefixes and corresponding target values in the majority class were randomly undersampled to contain an equal amount of prefixes as the least occurring class. Accordingly, the *RandomUnderSampler* from the Imbalanced learn library was used in Python (The imbalanced-learn developers, 2022c). Next, the the prefixes and matching target values for the two classes containing the most extreme values were randomly oversampled in Python using the *RandomOverSampler* (The imbalanced-learn developers, 2022b), after which the classes contained as many samples as the old majority class contained. Instead of using directly the prefix feature vectors for X in the *RandomOverSampler* and *RandomUnderSampler*, the prefix indices were used. Seeing that these resampling functions cannot take 3d-arrays (each prefix is a 2d-array) as input, this was a better approach. After the resampling, the correct prefixes for in the under- or oversampled dataset could be retrieved using these indices. The resampling was only performed on the training dataset. This is especially important for oversampling, since when performed before splitting the dataset into a train, validation, and test set, there is a probability that data leakage between the different sets occurs. It could happen that prefixes which are duplicated in the dataset then appear

in both the train and validation set, which would give a bias in validation performance. Performance values would be higher than they should be since the model had to make a prediction for a prefix that it has already seen during training, which is easier.

5.1.2 Method Selection and Implementation

Within this master thesis five different models were selected to predict the difference in days between the contractual delivery date and actual delivery date for PO items, each having a different neural networks architecture. These five models were based on findings in recent literature indicating potential high performance: MLP (Venugopal et al., 2021), GCN (Venugopal et al., 2021), LSTM (Tax et al., 2017), CNN (Pasquadibisceglie et al., 2020), and BIG-DGCNN (Chiorrini et al., 2021). The next paragraphs will further clarify the reasons for selecting the different models and what steps have been taken in the implementation of the models.

MLP

The MLP is in fact the most basic feed forward neural network, as explained in Section 2. This model was selected although being a basic neural network, since it can learn complex patterns due to multiple layers of neurons (not only linear relationships) (Fath et al., 2020). Besides, Venugopal et al. (2021) highlight that the MLP is a strong baseline which frequently outperforms more sophisticated NNs like the CNN, LSTM, and GCN. Furthermore, the P2P process, from which the data is used as input, is process where the majority of process instances can be described by a simple process model. These process characteristics might indicate that such a basic neural network model could be able to effectively learn the relations between input features and target values.

The MLP was implemented using the Pytorch-library within Python (PyTorch Contributors, 2022a). Although the implementation was based on the approach of (Venugopal et al., 2021), their code had to be adjusted to make the model work for the defined prediction task and on the selected input data. A detailed description of all the modifications applied to the code is given in Appendix C. Furthermore, a summary is provided of the MLP architecture in Table 31.

Table 31: MLP architecture configuration

Setup	Explanation
Prefix feature vector	Shape of (n_unique_activities, n_features) for each prefix
Sequence encoding	Prefix-padding
Activation function	ReLU-activation between the layers, no activation in the final layer
Optimizer	Adam

GCN

The main idea behind the GCN is that the model combines the graph representation of the process model discovered from the event log with the features for each event of the traces, in a graph convolutional layer (Rama-Maneiro et al., 2021b). Within the GCN version of Venugopal et al. (2021) the type of graph to represent the process is a Directly Follows Graph for the entire event log. The graph consists of nodes and edges connecting the nodes. The nodes correspond to the set of events in the process, the edges to possible sequences of activities. The graph is defined by an adjacency matrix, which represents the control-flow. Besides the adjacency matrix, the GCN layer needs another input which is matrix X containing the feature vectors for every node in the graph. Each prefix is then assigned a matrix X where the rows represent the feature vectors for the nodes in the graph. Hence, the number of rows equals the number of unique activities in the used dataset. The number of columns in X corresponds to the length of the feature vector. The rows in matrix X are then filled for the activities corresponding to the events that have occurred in the case up until the certain prefix length. In scenarios where an event corresponding to a particular activity has occurred more than once in a case, only the feature vector for the most-recent occurrence of that event is stored in matrix X. Whenever an event corresponding to a particular activity has not occurred yet in a given case, the feature vector for that activity is filled with zeroes (Venugopal et al., 2021).

The GCN architecture was selected based on the fact that the process structure is exploited by this architecture. Since this type of model relies on process models as input to obtain information from the connections between model activities, behavioral patterns typically seen in process models, e.g. loops and parallels, can be better detected (Rama-Maneiro et al., 2021b). Structures like loops, parallels, and choice are flattened in the event log, since traces only record the sequence of executed activities. Recognizing these higher-level constructs can be beneficial for the predictor and improve predictive performance (Chiorrini et al., 2022). Moreover, it is discovered in Philipp et al. (2019) that the GCN could converge faster to a better regression value than a basic NN with for example only fully connected linear layers.

Alike the MLP, the GCN was implemented within Python using the Pytorch-library since it was based upon the approach of (Venugopal et al., 2021). Since the code from both the MLP and GCN were based upon the approach of Venugopal et al. (2021), the same modifications were applied to the code for the GCN as for the MLP (see Appendix C). However, one additional adjustment needed to be implemented specifically for this NN architecture. This particular adaptation is explained in detail in Appendix C. Lastly, a summary is provided of the GCN architecture in Table 32.

Table 32: GCN architecture configuration

Setup	Explanation
Prefix feature vector	Shape of (n_unique_activities, n_features) for each prefix
Sequence encoding	Prefix-padding
Adjacency matrix	Laplacian matrix, where a Laplacian transformation is applied to a binary adjacency matrix
Activation function	ReLu-activation between the layers, no activation in the final layer
Optimizer	Adam

LSTM

This type of NN is an approach to sequence modeling problems, and falls under the RNN form of a NN. A RNN is unfolding and each step in the unfolding refers to a time step, where x_t is the input at time step t. The NN takes a sequence of arbitrary length as input by giving the NN a feature representation of one element of the sequence at each time step. The hidden state at time step t, h_t, contains information of all time steps up till t, and is updated after each time step with information of the new input. The output at time step t is hidden state h_t multiplied with a weight. The LSTM is a special form of RNN where the main distinction is that h_t is learned by means of a more complex memory cell (Tax et al., 2017). The choice for the LSTM architecture was based upon multiple reasons. First of all, it is one of the most popular architectures in the field of PBPM (Rama-Maneiro et al., 2021a). Besides, due to the temporal organization of the activities in a trace researchers often prefer RNN architectures since they are designed to deal with temporally dependent data. The LSTM is a special RNN architecture that can model long-term dependencies in a powerful way (Tax et al., 2017).

The LSTM was implemented in Python using the Tensorflow-library, since it was based upon the implementation approach of Tax et al. (2017). Again, specific changes had to be made to the LSTM to be able to predict the delivery date from the selected input data. All the changes are described in detail in Appendix C. Furthermore, a summary is provided of the LSTM architecture in Table 33.

Table 33: LSTM architecture configuration

Setup	Explanation
Prefix feature vector	Shape of (max_trace_length, n_features) for each prefix
Sequence encoding	Prefix-padding
Activation function	Tanh activation and Sigmoid recurrent activation in LSTM layers (between states of an LSTM (Tax et al., 2017)), no activation in the final layer
Optimizer	Nadam
Dropout	In all LSTM layers
Batch normalization	After each LSTM layer

CNN

The CNN is an extension of the basic fully connected feed-foward NN by adding three concepts: convolution, pooling, and weight sharing (Pasquadibisceglie et al., 2019). The model input is grid-like, and in the case of Pasquadibisceglie et al. (2020) a 2D-image. These images are extracted from traces in the event log. Subsequently, the input is given to the feature extraction layers (pairs of convolution and pooling layers), and then passed to a fully connected layer, providing the output. The convolution concept is a set of filters sliding along the whole input grid to process small local parts of the input (applying weights to the input). A pooling layer is added to reduce dimensionality of the resulting feature map by highlighting the most important features. The CNN was selected since it has an efficiency advantage of faster training and inference (Rama-Maneiro et al., 2021a). Moreover, this NN has shown exceptional ability to understand the spatial structure of an input. Pasquadibisceglie et al. (2019) indicates in his paper that in recent literature CNNs promise very high accuracy whenever applied to image data embedding a clear spatial structure. Therefore, it is worthwhile to investigate the application of the CNNs to non-visual sequential data from the traces of event logs. Converting temporal event data to spatial data may result in a reasonable representation of trace data which may be used for accurate predictive analytics.

The CNN was implemented in Python using the Tensorflow-library, since it was based upon the implementation approach of Pasquadibisceglie et al. (2020).
Interestingly, Pasquadibisceglie et al. (2020) applied min-max normalization such that the value of each feature ranges in $[0.0, 1.0]$. Equation 5 shows how the transformation was performed. Pasquadibisceglie et al. (2020) applied this transformation since it converts all features values into the same range, through which the same weight is assigned to the independent features (Pasquadibisceglie et al., 2020).

$$z = \frac{x - min(x)}{(max(x) - min(x))} \tag{5}$$

Considering it as a reasonable choice from Pasquadibisceglie et al. (2020), the min-max normalization was then performed on top of the already applied log-scaling of the input features. The normalization was only performed for the input features and not for the target values.

Furthermore, several changes were performed to make the CNN of Pasquadibisceglie et al. (2020) work for our prediction task and dataset. A detailed explanation of these modifications is provided in Appendix C. Lastly, a short overview of the CNN architecture is given in Table 34.

Table 34: CNN architecture configuration

Setup	Explanation
Prefix feature vector	Shape of (1, n_features) for each prefix
Sequence encoding	Aggregation encoding
Activation function	ReLU-activation between layers and no activation in the final layer
Optimizer	Adam
Dropout	Two times, after a dense layer

BIG-DGCNN

The BIG-DGCNN has a lot of similarities with the GCN, however, also notable differences. One of

these differences is that another approach for the input graph is used. The traces are converted into instance graphs using the Building Instance Graph (BIG)-algorithm proposed in Diamantini et al. (2016) and enriched with feature information. These are given as input to the model. Moreover, several different type of layers are used.
One of the reasons for selecting this architecture was that the process structure is taken into account when computing the time features such as time between an activity and its predecessor. This means the causal predecessor of the activity is considered, instead of the one occurring immediately before in the sequence. It is argued that this is a more accurate way of computing time intervals which can better support the predictor in detecting temporal relations among activities. Moreover, this NN works especially well in the presence of parallelism in the dataset.

The implementation of the BIG-DGCNN was inspired by the approach of Chiorrini et al. (2022). Therefore, the BIG-DGCNN was developed using the Pytorch-library within Python (PyTorch Contributors, 2022a). However, before the BIG-DGCNN could be implemented, it is important to specify that special data preparation steps were required for this NN type. First, artificial start and end events were added to each case using the "Add Artificial Events" plugin from Jan Claes (Claes, Jan, 2022) within process mining tool ProM (Van Dongen et al., 2005). It ensures that potential parallelism at the end or start of a process execution is properly modeled (Chiorrini et al., 2022). To be able to use the ProM plugin, the event log had to be converted from a CSV-file into a XES-file. For this, the log conversion function from PM4PY in Python was applied (Fraunhofer FIT, 2021a). Next, a process model, in the form of a Petri net, was needed for the BIG-algorithm. Hence, the Petri net resulting from best the data analysis phase (see Section 4.2), could be used but with slight adaptations. The discovery algorithm used to retrieve this Petri net seemed to be the same as used by Chiorrini et al. (2022). Making use of the functions from the *petri _utils* module of PM4PY, a start and end transition as well as a start and end place could be added to the Petri net. Moreover, arcs were changed accordingly (Fraunhofer FIT, 2021b). Thereafter, all the required input for the BIG-DGCNN was acquired.

Subsequently, the code regarding the implementation of the approach Chiorrini et al. (2022) had to be modified to make it work for the amount of data that was used for training, for the specified target, as well as for the selected features. For example, one of the problems with the implementation that occurred was that the time feature engineering and graph enrichment procedure as designed in Chiorrini et al. (2022) was extremely time consuming for the P2P dataset, even for the smaller subsamples. For example, when running the procedure for the subsample $k = 6$, it would run over 40 hours without any produced result. Therefore, the procedure was redesigned in Python. After redesigning the procedure the feature engineering and graph enrichment could be performed within 16 minutes for the subsample $k = 6$.
The explained steps in Appendix C highlight the most important changes performed to the original approach. These changes have been classified into changes related to the graph enrichment, the graph creation, and the model.

Lastly, an overview is provided of the BIG-DGCNN architecture in Table 35.

Table 35: BIG-DGCNN architecture configuration

Setup	Explanation
Prefix feature vector	Shape of an instance graph with feature attributes for each prefix
Sequence encoding	Instance graph encoding
Activation function	ReLU-activation between layers and no activation in the final layer
Optimizer	Adam
Dropout	One time, just before the final layer

Parameter Choice Neural Networks
Due to the long training process no hyper parameter optimization was performed in the end for the NNs. The Tree-structured Parzen Estimator approach was implemented for the different models,

however, not used. Since the smallest subsample that would be possible to make would consist of already 67486 cases (with the unique sampling method), and thus even more prefixes (maximum of 9 times the number of cases as this is the maximum trace length), it would always take extremely long to do hyper-parameter optimization. It would require the models to run multiple times for different parameter settings to be evaluated, however, one training run took already multiple hours for most models. Therefore, the parameter values have been based upon the values that have been used in the papers from which the models originate.

Random Forest

During the training and evaluation of the five NNs it was found that their performance was negatively affected by imbalanced data. It was observed that rare and extreme values for the target variable (difference in days between actual and contractual delivery date) had to be predicted from input data. This increased the average prediction error over all the prefixes.
Therefore, another approach was tested to investigate whether it could improve predictive performance. The goal of the approach was to first classify the extent to which the PO item would be delivered late/early, followed by predicting the exact delivery date with the best performing already developed NN. However, this NN would then be trained specifically on the cases within a class as used by the classifier, for each possible class. It was chosen to use a Random Forest (RF) as classification model. The next paragraphs elaborate on the reason for selecting the RF as classifier and how different versions of the RF were implemented.

The concept of the Random Forest (RF) is that it creates multiple decision trees from the data and averages their results to output a final prediction. Each of the decision trees is trained on a random sample drawn from the training dataset with replacement (Klinkmüller et al., 2018). With Random Forest Classification the final prediction for a class is the class that is most often resulting as output over all the decision trees. This concept is well depicted in Figure 10. The Random Forest Classifier was chosen since it is a popular method in the field of ML, and generally attains a better prediction accuracy than other ML models used for predictive process monitoring (Teinemaa et al., 2019; Verenich et al., 2019). Moreover, it has easy to understand hyper-parameters, prevents overfitting by aggregating over the results of all decision trees, is robust to outliers due to using multiple decision trees, and can estimate feature importance (Vadapalli, 2021). The choices made regarding the classes that the RF had to predict are explained in the next paragraph.

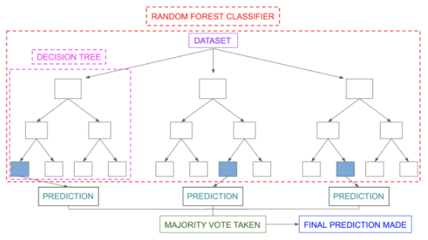


Figure 10: Concept of the Random Forest Classifier (Mbaabu, 2020)

Classes To Predict

First, it was investigated how the RF would perform when it had three classes to predict. These classes were determined by using the *qcut*-fuction from Pandas in Python (NumFOCUS, 2022d). This function discretized the target variable into three equal-sized bins in terms of percentage of data in it. These bins were based on the sample quantiles. The resulting bins were $(-69.001, -8.0]$, $(-8.0, 0.0]$, and $(0.0, 59.0]$. It was expected that having an equal percentage of data in each of the classes would help the RF recognize the different classes and lead to good performance (see Table 56 for the results). Moreover, RF performance was checked with only two bins to classify: too late (target $\in (0, 59]$) and on-time (target $\in (-69.001, 0]$). It could be argued that this would improve performance, since the RF has one class less to predict.
During discussions with the supply chain manager of Company X it was indicated that for Company X it was most important to predict the exact delivery date for items that would arrive too late, rather than for items that would arrive earlier than the contractual delivery date. Since both of these two classification models have a specific class for the too late cases, the models would help with the predictive task. If RF performance is well enough, they could predict whether a

PO item will arrive too late. The best NN could then be trained only on the cases with a target value $\in (0, 59]$, and predict the exact delivery date only for these type of cases since these are most important to Company X. This would reduce the range of values that the NN would have to predict which might improve its predictive performance. In order to test this, experiments were conducted with training and validating the NNs only on the too late cases (see Tables 50-53 for the results). Subsequently, classification performance was tested by developing a Random Forest to classify a PO item as being delivered very early (target value $\in (-69.001, -20.0]$), in a normal range (target value $\in (-20.0, 20.0)$, or very late (target value $\in [20.0, 59.0]$). These classes were determined based on domain-knowledge. During discussions with the supply chain manager of Company X it was not only indicated that it was most important to predict whether items would arrive too late, but more specifically it was mentioned that no exceptionally exact delivery date needed to be predicted for items of which it was sure that they would arrive more than/equal to 20 days late. Therefore the too late class was narrowed to the range of 20-59. In order to keep the ranges in similar width, the normal range was defined from 20 days early to 20 days late. This was as well substantiated by looking at the distribution plot of the target values in the total dataset, where it was seen that the central part of the values was in this range. In case that the RF performance is approved, the model could predict whether a PO item will arrive in the normal range. The best NN could then be trained only on the cases with a target value $\in (-20, 20)$, and predict the exact delivery date only for these type of cases since these are most important to Company X. This would reduce the range of values and amount of extreme values that the NN would have to predict which might improve its predictive performance. For the purpose of testing this assumption, experiments were performed with training and validating the NNs only on the normal cases (the results are outlined in Tables 50-53). The subsequent paragraphs go into detail about the RF implementation.

RF Implementation
All RFs were implemented within Python. The RFs did not only differ in terms of bins to classify, but additionally in the type of Random Forest classifier, type of hyper-parameter optimization that was done (e.g. number of trials, objective, parameter ranges), and the type of cross-validation that was performed.

Hyper-parameter optimization was implemented in Python with the Hyperopt-library (Bergstra et al., 2013). In order for Hyperopt to work the following elements needed to be defined: (1) an objective function that checks the relation between parameter values and the model's selected performance measure, (2) a configuration space over which the Hyperopt can do the search, (3) a search algorithm, and (4) a trials object where intermediate results are saved (Sukumar, 2020). The algorithm selected to perform the hyper-parameter optimization was the Tree-structured Parzen Estimator (TPE). The reason is that a Bayesian optimization approach which is more efficient than for example grid search or random search (Databricks, 2022). In consequence, a higher number of hyper-parameters and and wider ranges can be explored. The objective function of Hyperopt can only minimize. Since the performance measures used to evaluate the RF, like accuracy, precision, recall, and F1, all required to be maximized, the negative of the metrics were minimized. Within the configuration space the following hyper-parameters were present: number of estimators, maximum depth, maximum features, minimum samples in the leaf nodes, and minimum samples for splitting. The ranges for the hyper-parameters over which to do the search were initiated and subsequently increased or decreased based on whether the chosen values by the algorithm were close to the upper or lower boundary of the ranges. The initial ranges were based on best practices and the size of the dataset. For example, the max depth value was taken not too high to reduce computational time (range of 10-45), however, fairly high values for the number of estimators were chosen (range of 300-600) as more trees can learn the data better and the dataset is large (Mohtadi, 2017). Moreover, the minimum samples leaf value was chosen to be at least bigger than 1 to prevent overfitting, but not too large since that could lead to underfitting (Mohtadi, 2017).

Furthermore, 5-fold cross validation was first applied in order to prevent overfitting the data. However, since RF performance was affected again by imbalanced data it was decided to use stratified

5-fold cross validation instead. Normal K-Fold Cross Validation is not appropriate in the presence of imbalanced data because it randomly divides the data into k-folds. Therefore, the folds might likely have little or no data from the minority class bringing about a highly biased model (Huh, 2021). Stratified K-fold cross validation uses stratified sampling to obtain the folds. This ensures that the data is split randomly, while keeping the same imbalanced class distribution as in the complete training dataset, for each subset. During the split procedure of the stratified k-fold method, it was important to ensure that the different folds contained only complete cases, and no data leakage would occur due to prefixes from one case being present in multiple folds. Therefore, the training dataset consisting of prefixes was not directly used as input to the *StratifiedKFold.split*-function from Sklearn within Python (scikit-learn developers, 2022d). Instead the case IDs were used to perform the split on, so splitting was performed on a case-level rather than prefix-level. Subsequently, the prefixes belonging to the different folds could be traced back using these case IDs.

To further handle the problem of imbalanced data, various strategies were tested. Firstly, the RF classifier from Sklearn was evaluated, since it has the ability to incorporate class weights into the model (scikit-learn developers, 2022a). The "balanced_subsample" class weight was used which puts weights on the different classes based on its proportion in the bootstrap sample for each of the trees grown (scikit-learn developers, 2022a). Accordingly, it penalizes misclassifying the minority class. Moreover, the Balanced Random Forest from the Imbalanced learn Python library was examined. This RF type randomly undersamples each boostrap sample to balance it (The imbalanced-learn developers, 2022a). Furthermore, the RF classifier from Sklearn together with two types of random resampling strategies were tested. One strategy was randomly oversampling the minority classes such that for all minority classes the size became equal to one third of the training dataset, and subsequently undersampling the majority class to also contain one third of the training dataset. It was expected that an equal percentage of data in each of the classes would help the RF recognize the classes and therefore improve performance. The second strategy was to only randomly oversample the minority classes such that their size increased to 60% of the majority class. The increase of samples for the minority classes could help the classifier recognizing these classes. The strategies were implemented in Python using the *RandomOverSampler* and *RandomUnderSampler* (The imbalanced-learn developers, 2022b,c). These resamplers could be applied directly on the prefix data and target value data, since the arrays were 2-dimensional. The resampling was performed on the training dataset within each fold separately, to ensure that there was no data leakage as might occur if the oversampling would be performed prior to the cross-validation.

Finally, the multiple Random Forests were build, tested, and evaluated to examine whether the performance would be good enough to further build on this approach. The results are summarized in Table 56 in Appendix B.

5.1.3 Evaluation

The successive phase in the research framework is the evaluation. This phase looks at whether the business goals as specified by Company X can be achieved by implementing the PBPM method or not. When multiple methods are able to reach the business goals, the one having the best performance and efficiency in terms of time and cost will be the most appropriate method for the project. The performance of the different methods was evaluated by using multiple metrics. The next paragraphs explain the metrics selected to evaluate the various predictive models. A distinction was made between the NNs and the RFs, since the first are regression models and the second classification models.

The NNs have been evaluated using the following criteria:

1. *Mean Absolute Error (MAE).* This measure is the average absolute difference between the predicted value and the actual value. It is calculated with the following formula: $\frac{\sum_{i=1}^{N}|y_i - \hat{y_i}|}{N}$, where N is the sample size, y_i is the actual value and $\hat{y_i}$ the predicted value (Rama-Maneiro

et al., 2021a). The MAE was selected as evaluation criterion since it is a useful relative measure to compare forecasts for the same dataset across different models (Anaplan, 2022). A lower MAE value indicates that a predictive model is more accurate (Anaplan, 2022). For this metric a 95 %- confidence interval was calculated as well in order to express the certainty of the calculated mean absolute error. The 95 %- confidence interval could be calculated using the normal distribution. Since the size of the different datasets was for all bigger than 30, it could be assumed that the sampling distribution of the sample mean was normally distributed (Curran-Everett, 2017).

2. *Root Mean Square Error (RMSE).* This metric resembles the standard deviation of the prediction errors, i.e. residuals. The residuals are a measure of the distance of data points from the regression line, i.e. how close the actual values are to the predicted values. The RMSE value shows how spread out these residuals are. The value for the RMSE is computed using the following formula: $\sqrt{\frac{\sum_{i=1}^{N}(y_i-\hat{y_i})^2}{N}}$, where N is the sample size, y_i is the actual value and $\hat{y_i}$ the predicted value (Glen, 2022b). A model with a lower RMSE value produces more accurate predictions (Anaplan, 2022). This metric was added to the evaluation criteria as it is especially helpful in identifying the impact of outliers on predictive performance of the model (Anaplan, 2022). Since the predictions errors are squared before being averaged, the RMSE assigns a relatively high weight to large errors.

3. *Symmetric Mean Absolute Percentage Error (SMAPE).* This is a model accuracy measure based on percentage errors, meaning it is investigated how much the model predictions are deviating from actual values in percentages. Considering it is percentage-based, this metric could to be used to compare forecast performances between datasets (Anaplan, 2022). It has a lower and upper bound and the values range between 0 and 200%. The SMAPE value is calculated using the following formula: $\frac{100}{N}\sum_{i=1}^{N}\frac{|\hat{y_i}-y_i|}{(|y_i|+|\hat{y_i}|)/2}$. Seeing that the target variable can take the value zero (when the actual delivery date is equal to the contractual delivery date), this SMAPE had to be used as percentual error measure instead of the MAPE. Otherwise, problems would occur due to division by zero.

4. *Earliness.* Earliness was measured by evaluating the NNs for each prefix length. The improvement of prediction accuracy as the prefix length increases gives an implicit notion of earliness. Specifically, the smaller the prefix length when an acceptable level of accuracy is reached, the better the model performs in terms of earliness. This means that a model performs better with regard to earliness when less information is needed about the proceeding of a case for the model to make a sufficiently accurate prediction of its outcome. Earliness can be defined as a metric by the following: the smallest prefix length where the model achieves a specified accuracy threshold (Teinemaa et al., 2019). The accuracy measure that was used to evaluate performance on the different prefix lengths was MAE.

5. *Training Efficiency.* The training efficiency is a measure that expresses how fast a predictive model is trained and how fast it is converging to a certain level of accuracy. This was measured by the time that was needed for the model to perform one epoch of training and validation, with hours per epoch as unit of measurement.

In order to understand the construction of the different performance measures used for the Random Forest classifiers, it is important to explain the concept of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) first. The different performance measures are namely based on these concepts. The explanation is given in Table 36 and is based on the systematic review of Harane and Rathi (2020).

Table 36: Prediction categories (Harane and Rathi, 2020)

Concept	Explanation
TP	Events with positive event type classified correctly
FP	Events with negative event type classified as positive (wrongly)
TN	Events with negative event type classified correctly
FN	Events with positive event type classified as negative (wrongly)

When there are more than two classes to predict, e.g. a class A, B and C, instead of a positive and negative class, this is called multi-class classification. In that case the concepts of Table 36 are evaluated for each class. For example, a true positive for class A is then an instance of class A which is correctly predicted as class A, whereas a false positive for class A is an instance that is predicted to be of class A but is actually from another class.

The Random Forest classifiers were evaluated using the later evaluation criteria:

1. *Accuracy.* This is the quantity of overall correct predictions (for all the classes) over all the predictions that have been made, and is thus calculated by $\frac{TP+TN}{TP+FP+TN+FN}$ (Harane and Rathi, 2020).

2. *Precision.* This measures the proportion of the total amount of predicted instances that is predicted correctly. This can be calculated with the formula $\frac{TP}{TP+FP}$ (Harane and Rathi, 2020).

3. *Recall.* It describes the percentage of correct positive predictions made out of all positive predictions that have been made, and is calculated by $\frac{TP}{TP+FN}$ (Harane and Rathi, 2020). For example, if a model needs to make predictions for class A, B, and C, the recall for class A is the percentage of correct predictions of class A over all predictions that have been made for class A.

4. *F1-score.* This is the harmonic mean of precision and recall. Its value is computed by $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$ (Harane and Rathi, 2020).

A note must be made on precision, recall, and F1-score. Since most RF configurations had more than two classes to predict, the value for each of these measures was first calculated for the classes separately, and subsequently averaged over all classes to say something about the model in general. It was found that macro-averaging is the default aggregation method for precision, recall, and F1, for single-label multi-class problems (Peltarion, 2022). This is the case for our prediction task, since every instance must be classified into one (and only one) class. A PO item can for example not be arriving too late as well as too early. Moreover, since the dataset that was used for training, validating, and testing the RF models was imbalanced, it was best to use macro-averaging. When macro-averaging, all classes contribute equally regardless of how often they appear in the dataset (Peltarion, 2022). Therefore, this metric is insensitive to the imbalance of the classes.
For example, when using the micro- instead of macro-averaged precision as a metric to maximize during hyper-parameter optimization of the RF, the focus is more on optimizing the precision value for the majority class than for the other classes. Within the calculation of the average precision the precision for each class is weighted by its proportion in the dataset. For example, if the majority class represents 70% of the data, the average precision becomes 70% $*$ precision of majority class plus 30% $*$ precision of minority classes. Therefore, it is suggested that macro-averaging is more useful when having imbalanced data as it weights each class equally.
Furthermore, macro-averaging is more useful than micro-averaging for evaluation purposes since micro-average precision, recall, and F1, as well as accuracy are all equal when having a single-label multi-class problems. This becomes visible when inspecting the formulas for precision and recall. The numerators are the same, and every false negative for one class is a false positive for another class, making the denominators the same as well. Next, if precision equals recall, the mean of the

two values, i.e. F1, will be equal as well (Shmueli, 2019).

Besides, these evaluation metrics, the supply chain manager of Company X was asked to indicate an order of what type of incorrect predictions would have the most negative impact on their expediting process. The types of wrong predictions were determined based on the confusion matrix and thus divided into six categories: 1-6, see Table 37.

Table 37: Confusion matrix: RF with domain-knowledge based bins

Actual Class/Predicted Class	$(-69.001, -20]$	$(-20, 20)$	$[20, 59]$
$(-69.001, -20]$	TP	1	2
$(-20, 20)$	3	TP	4
$[20, 59]$	5	6	TP

Category 1 means for example that a PO item which would actually be delivered between 69 and 20 days before the contractual delivery date is predicted to be arriving between 19 days before and 19 days after the contractual delivery date. Moreover, category 3 means that a PO item which in fact would be delivered between 19 days before and 19 days after the contractual delivery date, is predicted to be delivered between 69 and 20 days before the contractual delivery date. The supply chain manager indicated that it should not be problematic if wrong predictions of category 1, 2, and 4 occur. However, incorrect predictions of category 3, 5, and 6 definitely should be avoided, where category 5 has the most negative impact, followed by category 6. Therefore, the best performing RFs in terms of the listed evaluation criteria could be further evaluated by inspecting the proportions of incorrect predictions of category 3, 5, and 6 happened. For example, for category 3 this would be calculated by dividing the number of predictions in box 3 with the sum of all predictions made for actual class $(-20, 20)$ (the correct and incorrect predictions).

5.1.4 Deployment

Finally, the predictive model which was evaluated to perform the best, was implemented within the Celonis environment of Company X. The predictive model could be implemented by using the integrated ML workbench of Celonis. This ML workbench allows users of Celonis to run a Python script within the platform on data embedded into it and to push back the results to the data model, after which the results can be used within for example new analyses (Celonis Inc., 2022). The trained model was added to the ML workbench as well as a created Python script for transforming the data for the running cases into the format which the model could understand. The preprocessing script was only applied to the prefixes belonging to the cases as selected by the process filters specifically for implementation (see Section 3.6). The pseudocode with all the functions applied for transforming the data into the correct input using Python is given in Algorithm 2. After transforming the data, the predictions could be made by loading the model and using the *model.predict*-function from the Tensorflow library on the prepaired data (Tensorflow, 2022c).

Algorithm 2: Pseudo-code of preprocessing performed with implementation

```
def preprocess(df_in):
    df_out=df_in.copy();
    rename_dict={'BU': ''bu', 'TTM OTIF': 'ttmotif',...} ;
    df_out.rename(columns=rename_dict,inplace=True) ;          ▷ Rename necessary columns
    dropcols=[attributes which were not used for training];
    df_out = df_out.drop(columns=dropcols); ;                   ▷ Drop unnecessary columns
    return df_out;
def apply_hashing(df_in):
    df_out=df_in.copy();
    hash_dict= Load dictionary containing hash labels;
    bin_dict= Load dictionary containing bin names;
    for feature in list of all categorical features do
        df_out[feature]=df_out[feature].apply(lambda x: str(hash_dict[feature][x]) if x in hash_dict[feature].keys() else str(length of
         hash_dict[feature] +1) ;       ▷ The feature values are hashed by replacing the value with a string label
    end
    for feature in list of all categorical features do
        df_out[feature]=df_out[feature].apply(lambda x: x.replace(x,'others') if x not in bin_dict[i] else x) ;   ▷ The labels are changed
         into the name of the corresponding bin (dummy name)
    end
    return df_out;
def add_dummies(df_in):
    df_out=df_in.copy();
    dummy_cols=[categorical features] ;                         ▷ Columns that need one-hot encoding
    df_out=pd.get_dummies(df_out,prefix=dummy_cols,columns=dummy_cols) ;   ▷ Automatic function of Pandas creating dummies
     and removing original columns
    return df_out;
def add_missing_cols(df_in):
    df_out=df_in.copy();
    binsTL= open list of categorical feature columns as ordered in training set;
    for feature in list of all categorical features do
        bins=[all column names for feature as in training set];
        for element in bins do
            if element not in df_out.columns:
                df_out[element]=np.zeros(length of df_out) ;    ▷ Add column for element, filled with zeros
        end
    end
    return df_out;
def reorder_cols(df_in):
    df_out=df_in.copy();
    columnsTLimport=open list of columns present in the total preprocessed dataset;
    Remove columns from columnsTLimport that were not used as features for training ;
    newTitles=[column titles in order of features as during training];
    df_out=df_out.reindex(columns=newtitles) ;                  ▷ Change the feature column order in the input frame
    return df_out;
def normalize(df_in):
    df_out=df_in.copy();
    log_dict= Load dictionary containing binary for numerical features whether log(x+1) or log(x) was used for normalizing the feature;
    numlist=[all numerical features];
    for feature in numlist do
        if log_dict[feature] == 0:
            df_out[feature]=log(df_out[feature]+1)
        else:
            df_out[feature]=log(df_out[feature])
    end
    return df_out;
def get_prefixes(df_in):
    frames, df_out= frequency_encoding(df_in) ;                 ▷ Convert binary one-hot to frequency one-hot encoding
    Drop target column and add to the back of df_out;
    prefixes= aggregation_encoding(df_out,frames) ;  ▷ Create 1D-feature vector for each prefix combining df_out and frequency
     encoding frames
    prefixes= prefixes.groupby('caseID',sort=False,as_index=False).last() ;  ▷ Keep only the one prefix for each case, the one with
     most information (largest length)
    scaler = Load min-max scaler as was fitted on the training dataset; prefixes = scaler.transform(prefixes) ;   ▷ min-max normalization
     for all features within prefixes
    X_implementation = rgb_image_generation(prefixes) ;         ▷ Convert prefixes to RGB images
    X_implementation = np.asarray(X_implementation) ;           ▷ Convert images to arrays
    X_implementation=X_implementation.astype(float32) ;         ▷ Change the values to float data type
    X_implementation = X_implementation/255.0 ;                 ▷ Normalize the image pixel values to a range from 0 to 1
    return X_implementation ;
```

5.2 Predictive Process Monitoring Results

This chapter summarizes the results for the predictive process monitoring part of this thesis. Since Chapters 3 and 4 already elaborated on the outcomes for the first two phases of the research methodology (Figure 3), business understanding and data understanding, this section focuses on the successive four phases of the cycle: data preparation, method selection and implementation, evaluation, and deployment. The performance results for the different neural network architectures are given, and performance is analyzed, after which the different architectures are compared and a best network is selected. Moreover, the results for the RF classification models are examined.

5.2.1 Event log sampling

As already explained in Section 5.1.1, event log sampling was an important part of the data preparation phase, where the goal was to reduce the original dataset to a workable size for the predictive models and at the same time maintaining similar performance as when the original dataset was used. In order to explore the effect on computational time for training the predictive models, subsamples of different sizes were created from the entire data set. The exact procedure for the event log sampling is explained in Section 5.1.1. Four different divisors have been used to create four subsamples, different in size. These divisors, k, have a value 3, 6, 12, and 24. This means that for example in the case of $k = 3$, for each process variant one third of all cases belonging to the process variant were randomly selected. Moreover, a unique method has been explored where randomly one case was selected for each process variant. The values for number of cases and prefixes in each of the subsamples are provided in Table 38.

Table 38: Number of cases & prefixes in the various subsamples

Subsample	#cases	#prefixes
Subsample ‘Unique’	67486	451418
Subsample $k = 24$	104561	707808
Subsample $k = 12$	149064	1014710
Subsample $k = 6$	241986	1654756
Subsample $k = 3$	432465	2965694

In order to evaluate whether the created subsamples are a good representation of the total dataset, the distribution of the target variable is examined. The distribution of the target variable for the total dataset is shown in Figure 11, whereas the distributions for two types of subsamples are highlighted in Figure 62. The distribution plots of the target variable for all subsamples are given in Figure 62 in Appendix B.

Overall, it can be observed that the subsamples generated with the divisor method represent the total dataset quite well in terms of target value distribution. For example, a practically identical shape is noticed when comparing the distribution of subsample $k = 6$ with that of the total dataset. However, differences in terms of distribution can be noted when looking at the four k-subsamples. When k increases, the distributions seem to become more flat. For instance, with subsample $k = 3$ a peak in the plot is present for target value zero at a proportion of 0.058 and a second peak around target value -6 at 0.037, giving a difference of around 0.021. With subsample $k = 24$, these peaks are found at 0.042 and 0.030, giving a difference of 0.012 which is smaller. This indicates that the proportions of the different target values that are present in the dataset move closer to each other when k increases and thus subsample size decreases. Hence, the values for the target variable within subsamples with a higher k value are more equally spread throughout the whole range of the target values. This indicates that the subsamples with higher values for k represent the total dataset not as good compared to the subsamples with lower values for k.

The results for the unique subsample, consisting of only one randomly selected trace for each of the variants, are less promising. As depicted in Figure 12, the target value distribution looks very different from that of the total dataset. However, this is not unexpected. Due to keeping only one trace for each variant in the subsample, the frequencies are not preserved of the variants, and therefore neither for the feature and target values.

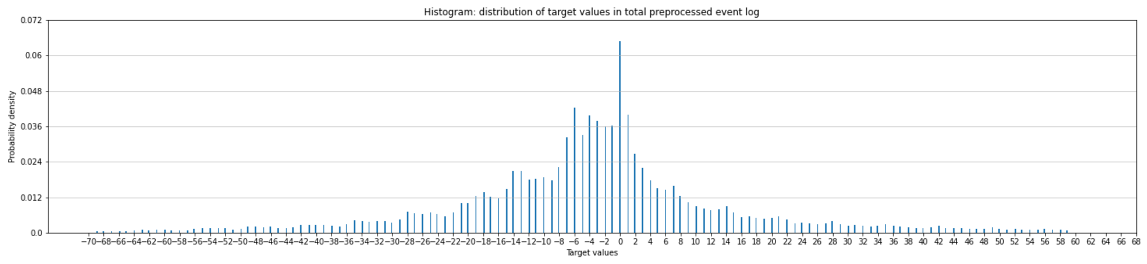


Figure 11: Distribution plot of the target variable for the total dataset

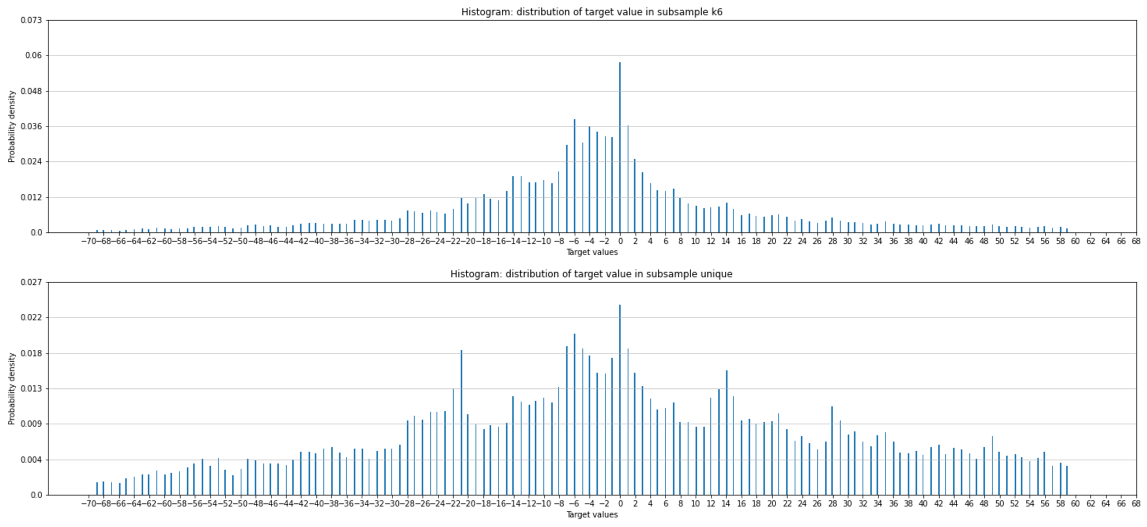


Figure 12: Distribution plot of the target variable for the subsamples

Hence, it can be concluded that the total dataset is best represented by the k-subsamples. To inspect the subsamples in more detail, the mean target values and corresponding 95%-confidence intervals are listed in Table 39.

Table 39: Mean target values and corresponding 95%-CIs for the various subsamples

Subsample	Mean target value	95%-CI
Subsample 'Unique'	-0.573	[-0.657,-0.489]
Subsample $k = 24$	-1.907	[-1.965,-1.850]
Subsample $k = 12$	-2.596	[-2.640,-2.552]
Subsample $k = 6$	-3.193	[-3.224,-3.161]
Subsample $k = 3$	-3.571	[-3.594,-3.549]
Total dataset	-3.842	[-3.855,-3.830]

At first sight, the distributions of the target values for the k-subsamples look fairly similar to that of the total dataset, but the 95%-CIs show that the mean target values are significantly different (Frost, 2019). However, the values for subsample $k = 3$ and $k = 6$ are fairly close to the values for the total dataset. But, to get more information about the entire distribution instead of only the mean, the cumulative distribution functions (CDFs) are plotted and depicted in Figures 63 - 67 in Appendix B. An example is given for subsample $k = 3$ and subsample $k = 6$ in Figures 13 and 14. Within these plots, for each of the subsamples the CDF of the total dataset is compared with the CDF of the respective subsample. It shows for each point of the x-axis (target variable) the percentage of data points that have an equal or lower value (Courthoud, 2022).

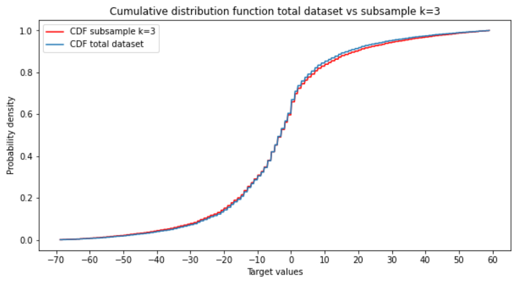


Figure 13: The CDF for the total dataset and subsample $k = 3$

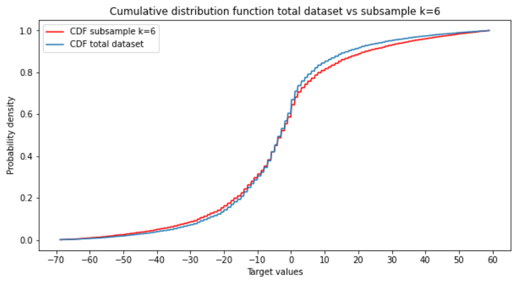


Figure 14: The CDF for the total dataset and subsample $k = 6$

When investigating all the CDF plots for the five subsamples, it can be concluded that the distribution of subsample $k = 3$ is most similar to the distribution of the total dataset, followed by subsample $k = 6$. The CDF-lines in Figure 13 for subsample $k = 3$ are almost entirely overlapping, showing that the proportions of the different target values in both datasets are very similar.

Based on these results, it can be concluded that the total dataset is best represented by subsamples $k = 3$ and $k = 6$. Since, the histogram distribution plots as well as CDF plots show that the distributions of the target variable for all k-subsamples are farily similar to that of the total dataset, it was decided to conduct all experiments for the predictive models with those subsamples. This will give more insight in the trade-off between computational efficiency and predictive quality for predictive models using the various subsamples. However, when performing the final comparison between the predictive models it is best to take the results for the same dataset. Moreover, since no statistical tests have been performed to prove that the subsamples are good representatives of the total dataset, it remains just an assumption. Therefore, when selecting the best predictive model the decision is based on the results of the experiments performed with the total dataset. These results are more reliable, as no bias can be created by slight differences in the target value distribution within a subsample.

5.2.2 Train-Validation-Test Split

Like described in Section 5.1.1, another important step in data preparation is splitting the total dataset into different datasets to be used for model evaluation. The procedure for splitting the total dataset into a train, validation, and test set has already been explained in Section 5.1.1. Here, we discuss the resulting train, validation, and test partitions after the splitting procedure, as well as the distributions of the target values within these datasets. An example is provided of subsample $k = 6$. Figures 15 - 18 show for subsample $k = 6$ the distribution of the target variable for the total dataset along with those of the train, validation, and test dataset.

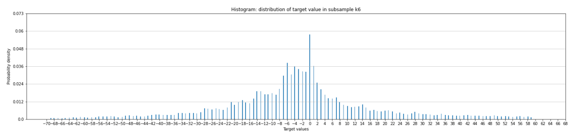


Figure 15: Distribution plot of target values within subsample $k = 6$

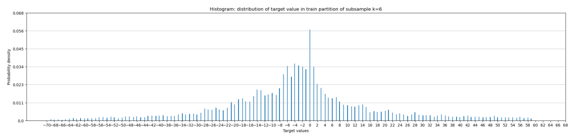


Figure 16: Distribution plot of target values within train partition of subsample $k = 6$

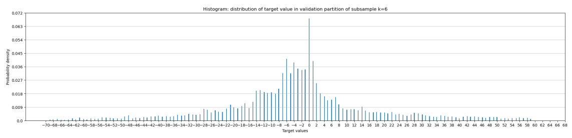


Figure 17: Distribution plot of target values within validation partition subsample $k = 6$

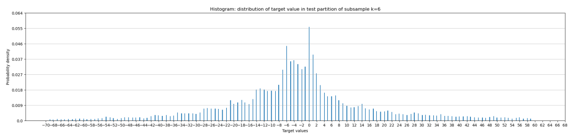


Figure 18: Distribution plot of target values within test partition of subsample $k = 6$

Clearly, all subsets of the dataset and the total subsample display a highly similar distribution of the target variable in terms of shape. The values are centered around the range of -20 and 20 where the tails stretch out to left and right of the outsides of this range. Moreover, no strong deviations are detected in the shape of the distributions among the train, validation, and test partition. This is an interesting observation, since the splitting procedure did not take the target distribution into account, and was solely temporally based (i.e. the training partition contains the earliest available data and the test partition the most recently available data). It could be an indicator that the P2P process has been relatively stable over the years in terms of when items were delivered compared to their contractual delivery date.
Since all subsamples have a similar shape (see Figure 62 in Appendix B) and a similar splitting procedure is used for all predictive models, it is assumed that there are no large differences between the target distributions of the different partitions for all subsamples and when used for the various models. With this knowledge, the predictive models can be safely trained, validated and tested on the different datasets. If the target values were dispersed differently in the train partition compared to the test partition, the model performance would not be reliable since a predictive model is not able to make good predictions when trained on different data than used for testing. The model would only predict values for the target variable in the range that the model has seen most during training, which would result in high errors if the target values in the test set are not in the range of the training set or not as frequently present as in the training set.

5.2.3 Predictive Models

This section describes the various experiments conducted for the multiple predictive models, and provides an analysis of the results.

Experiments
As described in the method selection and implementation section (Section 5.1.2 of the Methodology chapter) five NN models were selected for the predictive task. Therefore, these five different models have been tested to predict the difference in days between the contractual delivery date and actual delivery date for purchased items, each having a different neural networks architecture. These five models were based on findings in recent literature indicating potential high performance: MLP (Venugopal et al., 2021), GCN (Venugopal et al., 2021), LSTM (Tax et al., 2017), CNN (Pasquadibisceglie et al., 2020), and BIG-DGCNN (Chiorrini et al., 2021). In order to get a thorough evaluation of the different neural network types, the networks have been tested and evaluated on various sample sizes, multiple feature sets, distinct parameter configurations, different types of cases, and using numerous prefix encoding types. These experimental decision factors and their ranges are summarized in Table 49 in Appendix B.
The performance of the NNs is analyzed on a higher level concerning the entire dataset, as well as on a deeper level inspecting performance for different types of cases and for different prefix lengths. All experiments that have been performed for the separate NN architectures and their corresponding results are summarized in Tables 50 - 53 in Appendix B.

Moreover, as mentioned in Section 5.1.2, another approach involving a Random Forest has been tested as well to investigate whether it could improve predictive performance. The approach consisted of first classifying the extent to which the PO item would be delivered late/early, followed by predicting the exact delivery date with the already developed NNs which would then be trained on the specific late/early cases. Therefore, multiple Random Forests have been built, tested, and evaluated to examine whether the performance would be good enough to further build on this approach. The RFs differ in terms of the chosen classification bins, RF type, as well as the type of hyperparameter optimization and cross-validation performed. Furthermore, experiments have been conducted where the NNs were trained only on cases with a target value in a certain range corresponding to the chosen bins for the RF (see Tables 50 - 53). The decision factors and corresponding ranges for the RF experiments are summarized in Table 55 in Appendix B. All experiments for the RFs are listed in Table 56 in Appendix B.

Since multiple complex predictive models needed to be tested on very large datasets (see Table 10 in Section 3.6), the Dutch national supercomputer called Snellius has been used for all the experiments. This server was made available through the University of Technology Eindhoven. Snellius contains much faster processors and higher memory availability (SURF, 2022). For example, one CPU node on which a job (for a model) could be run contains 128 processor cores whereas the available Intel Core i7 vPro 8th Generation laptop contains only 4 cores (Intel, 2022). Moreover, it allows to run several models at the same time.

Snellius works with the workload manager SLURM. It allocates access to compute nodes to users for a specified duration, provides a framework for starting, executing, and monitoring jobs on the allocated nodes, and manages a queue of pending jobs on the server. Therefore, in order to run the predictive models that were created in Python, a separate job script has been created for all the models defining the duration for running it, the memory that needs to be allocated, the working directory, and specific Python model file among others. The batch system then knows which nodes from the server to allocate and for how long (SchedMD, 2021). To be able to easily change variable values for the models at the start of running a model, a configuration Python file was created for each of the models. These files contain arguments for the variables that need to be flexible, e.g. the number of epochs, batch size, learning rate, data directory, and checkpoint directory. A command can then be added at the end of a job script, using such an argument, to assign a new value to a variable. For example, adding the command '- -batch_size=256' changes the value taken for the batch size from default to this newly assigned number. To retrieve the arguments in the Python model files a self-built *load*-function was created in the configuration files. In this function an argument parser was created, to which subsequently multiple arguments were added (Python Software Foundation, 2022). When running the job script for a model experiment, such a configuration file is loaded into the model file where the arguments can be used to assign as a value to variables within the model files.

This section continues with elaborating on the performance analysis for each of the NN architectures separately, by comparing the results on the validation dataset for different configurations of the NN under investigation. Subsequently, the performance comparison across the different NN architectures is discussed by reflecting on their results for the test and validation dataset. Next, the best NNs are selected and examined in more detail in an effort to select the final best NN. Lastly, the results for the RF models are analyzed and it is investigated whether the RF approach can improve predictive performance.

NN Performance Analysis

In this subsection, each of the five NNs is analyzed independently. The results for the different experiments, as outlined in Tables 50 - 53 in Appendix B, are used to draw conclusions about the performance of the various NN configurations.

MLP

First, the performance of the MLP is analyzed using the results of the executed experiments as described in Table 50 in Appendix B.

Noticably, when the size of the subsample increases, the performance in terms of MAE improves.

For example, the MAE decreases from 13.786 to 10.217 when moving from subsample $k = 24$ to subsample $k = 6$. Especially, the unique subsample, which is the smallest in size, has unsatisfactory performance with a MAE value of 17.458. However, this was expected since the distribution of the target variable in the unique subsample is dissimilar to that of the total dataset, see Figures 11 and 12. Although performance is improving in terms of MAE when increasing the sample size, the training efficiency is dropping. The training time goes up from 0.225 hours per epoch to 0.373 hours per epoch when using subsample $k = 12$ instead of $k = 24$. It increases to even 0.497 hours per epoch when using subsample $k = 6$. Since training time was increasing a lot when increasing the sample size, thus a maximum of 41.723 hours was observed when full training was completed (100 epochs), no bigger sample size was tested than that of subsample $k = 6$.
Furthermore, log-scaling the target variable does not improve the performance in terms of MAE. The MAE rises from 10.217 to 19.773 when executing a log-scale transformation on the target variable. A reason could be that the MLP is less capable of recognizing the effect of the input on the output, as an effect of the decreased range of target values after the log-scaling.
Besides, implementing random oversampling of the most frequent target values and random undersampling of the extreme target values results in a slightly lower MAE value. The value decreased from 10.217 to 10.010 for subsample $k = 6$. The decrease is rather small, which might indicate that a more complex technique than random resampling is required to enhance performance.
Moreover, when applying embedding encoding for the prefixes, the MAE is not improved for the validation set. The value increased from 11.045 to 16.614 for subsample $k = 12$. A possible reason might be the following observation. When looking at the relation between the target values and absolute error values, it seems that in case of the embedding encoding the magnitude of the absolute error is related to the magnitude of the target value. The more extreme the target value is, either in negative or positive direction, the more extreme the MAE value is. This effect is present for the prefix padded encoding as well, however, of a less stronger degree. This is clearly visible in Figures 19 and 20.

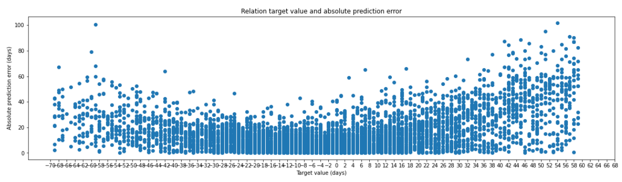


Figure 19: Relation between the target values and absolute error values for subsample $k = 12$ with the mainly used sequence encoding method - prefix padding

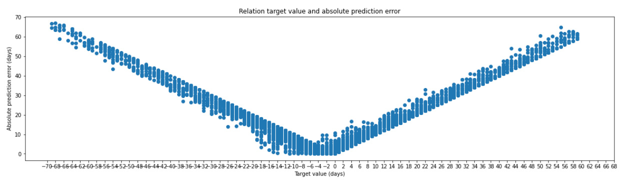


Figure 20: Relation between the target values and absolute error values for subsample $k = 12$ with embedding

Lastly, when using only the top features as input for the MLP, no obvious effect on MAE is found (see Section 4.3.4 for the list of top features). The value stays very similar for subsample $k = 6$, 10.696 days with the smaller feature set compared to 10.217 days with all selected features. Apparently, the features activity, weekday, number of changes to the delivery date, and vendor which are not included in the smaller dataset, are adding no evident value to the predictive quality of the model.

GCN
Next, the performance of the GCN is evaluated by inspecting the results for the different experiments as reported in Table 51 in Appendix B.

First of all, it is observed that the GCN has a fairly short training time of 0.155 hours per epoch when training the model on subsample $k = 6$. Especially compared to the MLP, for which the training time is 0.497 hours per epoch on the same dataset. Alike the MLP, log-scaling the target variable does not improve the performance in terms of MAE. When training the GCN on subsample

$k = 6$ and performing a log-scale transformation on the target variable, the MAE rises from 13.096 to 18.527 compared to when the target values are kept as-is. It is reasonable since the skewness for the target variable is actually small, with a value of 0.035 (see Table 20). Therefore, log-scaling the target might only just confuse the NN.
Furthermore, a similar result as for the MLP is found when using only the top features. It has almost no effect on the MAE value. The value for subsample $k = 6$ when using the smaller feature set (13.168 days) is very similar to when all selected features are used (13.096 days).

LSTM
Multiple experiments have been performed for the LSTM, which are summarized in Table 52 in Appendix B together with their results. Subsequently, the performance of the LSTM is evaluated.

Comparable to what is discovered for the MLP, the MAE is decreased when the sample size is increased. For instance, the MAE drops from 15.863 to 13.642 days when using subsample $k = 6$ instead of $k = 12$. An unexpected observation is that the training time decreased slightly when increasing the sample size, where the value went down from 0.100 to 0.071 hours per epoch when using subsample $k = 6$ instead of $k = 12$.
Increasing the number of neurons in the hidden layers from 100 to 300 neurons seems to have no real effect on performance in terms of MAE. Since for subsample $k = 6$ the MAE decreases from 13.642 to only 13.413 which gives a difference of only 0.229 days. Besides, it is noticed that the training time increased substantially from 0.071 to 0.215 hours per epoch.
Furthermore, alike for the MLP, implementing embedding encoding is not beneficial for the MAE value. A rather similar MAE value is noticed, since it becomes 15.857 instead of 15.863 days when using embedding for subsample $k = 12$. However, extreme overfitting is detected, where the training loss drops from 10.664 to 3.439 days over 100 epochs while the validation loss increases from 15.857 to 23.803 days. This effect is far less noticeable for the LSTM where regular prefix padding was used to encode the prefixes.
Again transforming the target values to log-scaled values is not beneficial for NN perfmance regarding MAE. However, the MAE value hardly changed whereas for the MLP and GCN a large increase is observed. The MAE value went up from 13.642 to 13.651 days for subsample $k = 6$.
Besides, implementing the random resampling stragegy like was done for the MLP decreases the MAE value to some extent. The value went down from 13.642 to 13.541. However, this decrease is so minor that possibly more complex methods are necessary to reduce the effect of imbalanced data.
Additionally, due to the short training time for the LSTM, experiments were executed with multiple smaller feature sets to check whether it would improve the MAE (see Table 52. However, for all smaller feature sets, an extreme increase in MAE value is observed. For example, when removing features activity, weekday, number of times delivery date has been changed, item, and vendor from the feature set, the MAE value is more than doubled, going from 13.642 to 30.808 days for the subsample $k = 6$. Furthermore, the resulting predicted values for the target are exceptionally high. When looking at the MAE values for the different type of cases for the LSTM with one of smaller feature sets it is confirmed that the LSTM seems to be better at predicting the late cases (MAE = 18.453) and that the high average error is because the model has become less good at predicting the early and on-time cases (MAE early = 38.308, MAE on-time = 28.775). Besides, the overfitting seems more extreme in this case, where over the epochs the training loss decreases from 10.784 to 9.239, whereas the validation loss increases from 30.808 to 34.269 days. The differences with regard to the MAE among the different smaller features sets are minor.
Lastly, the training period is often conspicuously short, where only between 11 and 14 epochs are performed even with early stopping which stops the training process after 10 non-improving epochs. Therefore, it seems that the model is not actually learning throughout the training process. When inspecting the training and validation loss values over the different epochs, it seems like overfitting is the underlying problem. For example, for subsample $k = 6$ the training loss decreases from 13.325 to 12.402 over the 13 epochs. On the other hand, the validation loss starts with a value of 13.717 days at the first epoch and ends with a value of 13.962 days at the last epoch. Even higher

values than for the last epoch are observed during in-between epochs. The same phenomenon is for example seen when the LSTM is trained with the log-scaled target values.

CNN
Various experiments have been performed to get an overview of the CNN performance on the validation set for various configurations of the CNN. The experiments and related results are listed in Table 53 in Appendix B. The next paragraphs elaborate on the analysis of the results.

First, experiments have been started with already a larger subsample, specifically subsample $k = 6$, since it seemed to yield best performance in terms of MAE for the MLP. Considering that the training time is relatively short for subsample $k = 6$, 0.166 hours per epoch, the CNN has been tested on the bigger subsample, subsample $k = 3$. Hereby, the MAE value is reduced with 0.807 days, since it decreased from 8.599 to 7.792 days. Seeing that the computational time has not increased extremely (0.288 hours/epoch), the CNN has even been tested on the entire dataset. This has led to an enormous increase in training time compared to subsample $k = 3$, seeing time goes up to 1.041 hours/epoch. Although the size of the dataset increased immensely, a decrease in MAE value of only 0.557 days is realized. Since the MAE value for this subsample comes closest to value for the total dataset, it can be concluded that subsample $k = 3$ is the best representative subsample of the total dataset. It reduces training time largely, while the MAE remains almost similar to that of the entire dataset.
Moreover, log-scaling of the target values increases the MAE value slightly, however, the effect is hardly noticeable. The MAE value rises from 8.599 to 8.954 days for subsample $k = 6$.
In addition, alike the MLP and GCN, using only the top features as input increases the MAE slightly. Subsample $k = 6$ shows a minor increase in MAE value when the smaller feature set is used, 9.228 days compared to 8.599 days.

BIG-DGCNN
Finally, the results for the BIG-DGCNN are analyzed based on the performance values reported in Table 54 for the multiple conducted experiments.

Only the results for the experiments using subsamples other than subsample $k = 12$ are considered in the analysis. It was found that while testing the BIG-DGCNN within the experiment using subsample $k = 12$, still an error was present in the code. Therefore, those results are not reliable. This error had nothing to do with the subsample, but was due to the following. It was noted that the target tensor had a different shape than the prediction tensor, for example the target tensor was a *torch tensor([23])* while the prediction tensor was a *torch tensor([23,1])*. Because the prediction tensor had a different shape than the target tensor, a problem arose with computing the loss. This resulted in getting the same prediction value for each input prefix. Therefore, the shape of the target tensor had to be adjusted such that it got the same shape as the prediction tensor. This was then done for all subsequent experiments. These experiments were performed using subsample $k = 6$ and the total dataset.

Again, for this NN as well, log-scaling the target value has almost no influence on MAE value. For subsample $k = 6$, the MAE value is decreased from 12.202 to 12.185 days. Furthermore, increasing the learning rate from 0.0001 to 0.01 has almost no effect on the MAE value. It increases slightly from 12.202 to 12.337 days. The training time decreases slightly from 1.950 to 1.659 hours/epoch. However, the learning process seems to have been negatively affected since training loss starts with a value of 13.404 days, decreases to 13.200 within 6 epochs and then increases again to 14.942 in the successive epochs. Moreover, the model converges faster to the best validation loss value seeing that only 14 epochs are performed until early stopping interrupts the training process. Knowing that the early stopping patience has a value of 10, this best validation loss was reached within 4 epochs.
Another interesting finding is that by adding the non-time features from the selected feature set (see Section 4.3.4) to the time features from Chiorrini et al. (2022) and giving this as input to the

BIG-DGCNN, the MAE value is decreased to a large extent. The value is reduced from 12.202 to 6.627 days which is almost half the error. However, it results in an extreme training time of 2.352 hours/epoch. But, if the batch size for training is increased from 64 to 256 the MAE value rises only slightly with 0.115 days, whereas training time is eminently decreased to 0.362 hours/epoch. This is an almost 2 hour decrease. This allowed to increase the maximum number of epochs for training from 30 to 100. By doing so, the model is training for more epochs, resulting in a MAE value which is only 0.017 days higher compared to the BIG-DGCNN trained on the batch size of 64. However, training time increased to 0.516 hours/epoch. A possible reason could be that by training the model for more epochs, more of the available memory of the processing node was utilized, making the training process slower.
Finally, the model was trained on the total dataset as well. The MAE value on the validation set decreases to a value of 5.584 day, which is around 1 day less compared to the resulting MAE values of the experiments performed with the same features in subsample $k = 6$. Although the size of the input dataset increased extremely, the MAE value did not decrease extremely. Since an increase in training time was expected due to the increase in size of the input data, the batch size for the training and validation set was set to 256. However, training time still increased immensely to a value of 2.918 hours/epoch. Since the MAE value barely changed for subsample $k = 6$ when increasing the number of epochs for training from 30 to 100, the BIG-DGCNN was only trained on the total dataset for 30 epochs. Moreover, it would take around 12 days to train the model if the model would train for 100 epochs.

The subsequent paragraph analyzes the NNs further by comparing the different architectures on a common subsample using multiple evaluation criteria.

NN Performance Comparison
In order to compare the performance of the different NN architectures, the results on the test partition of subsample $k = 6$ are used. This is the largest subsample on which all NNs could be tested within a reasonable amount of time. The performance for all five models is summarized in Table 40. All five models were trained, validated, and tested using the main feature set (see Section 4.3.4).
It is noted that the BIG-DGCNN performs best in terms of MAE on the test set with 7.119 days. However, it performs worst in terms of training time per epoch with 2.352 hours. The MAE value for the CNN is the second best, being approximately 2 days higher than the MAE of the BIG-DGCNN. However, the CNN performs much better in terms training time compared to the BIG-DGCNN as this is only 0.166 hours per epoch. This gives a saving of 2.186 hours for each epoch, which results in a lot of time when the NN is trained on multiple epochs. However, it is found for the BIG-DGCNN that when batch size is increased to 256 for the training set, the training time is reduced to 0.362 hours/epoch where performance on the validation set remains similar. Moreover, the BIG-DGCNN performs best in terms of RMSE. Since large prediction errors are given a relatively high weight in the RMSE score, it shows how outliers impact model performance. Because, the BIG-DGCNN has the lowest value for the RMSE, the model is least affected by outliers. The LSTM seems to be the fastest model to be trained where it takes only 0.0706 hours per epoch. However, its MAE value is the highest of all models. The same holds for the RMSE value, and therefore outliers have most effect on the LSTM's performance. In the end, the decision of which model is performing the best is a real trade-off between training efficiency, MAE, and RMSE. There are two models that stand out in terms of MAE and RMSE which are the CNN and BIG-DGCNN, where the BIG-DGCNN has the best values. However, the BIG-DGCNN's training efficiency is not as good as the CNN. Moreover, it takes more time for the BIG-DGCNN to produce the predictions compared to the CNN, with 0.317 hours compared to 0.0629 hours. To come to a final decision on one best model, these two models are more thoroughly evaluated in the next paragraph.

Table 40: Performance comparison on the test partition of subsample $k = 6$

Model	Specific parameter values	MAE (days)	RMSE	training time (hours/e-poch)	prediction time (hours)
MLP	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 1 run, 100 epochs.	10.658	16.125	0.497	0.0628
GCN	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 1 run, 100 epochs.	13.216	19.507	0.155	0.0676
LSTM	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, early stopping patience of 10, 1 run, 100 epochs.	14.108	20.674	**0.0706**	**0.056**
CNN	own feature set like specified above, 128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 1 run, 100 epochs.	9.176	15.153	0.166	0.0629
BIG-DGCNN	time features of Chiorrini et al. (2022) and own non-time features like specified above, learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, early stopping patience of 10, 1 run, 30 epochs.	**7.119**	**13.734**	2.352	0.317

Best Neural Network

The performance of the BIG-DGCNN and CNN, both trained on the total dataset, is further evaluated by closely examining the performance per prefix length, looking at MAE confidence intervals, and investigating the MAE relative to the target values. Hereby, it is determined which of the two models performs best and is most suitable for implementation within Company X. According to the senior supply chain manger of Company X, the goal for the performance of the NNs is a mean absolute error (MAE) of between 2-4 days, preferably 2 days.

First, the performance measures for the two models are listed in Table 41.

Table 41: Performance values for the test partition of the total dataset for the CNN and BIG-DGCNN

NN	MAE	95% CI	SMAPE	RMSE	Training time (hours/e-poch)
CNN	7.619	[7.606, 7.631]	91.108	13.110	1.041
BIG-DGCNN	5.867	[5.855, 5.879]	73.089	11.637	2.918

It is observed that the MAE value for the BIG-DGCNN is around 2 days less than the value for the CNN. Moreover, the 95% confidence intervals of both models are not overlapping. Therefore, it can be argued that the MAE of the BIG-DGCNN is significantly lower than that of the CNN (Frost, 2019). Moreover, the RMSE value is lower for the BIG-DGCNN. This means the BIG-DGCNN is less sensitive to outliers, and thus extreme target values, than the CNN. Furthermore, the SMAPE is lowest for the BIG-DGCNN. This would imply that the BIG-DGCNN generates the most accurate predictions. On the other hand, the training time per epoch is lowest for the CNN. Hence, the CNN performs better in terms of training efficiency.

To compare the performance of the two models in terms of earliness, the MAE values are checked for the different prefix lengths. Table 42 summarizes the performance of the CNN and BIG-DGCNN for each prefix length in terms of MAE, number of samples for the prefix length, and 95%- confidence interval for the MAE. Note that in the BIG-DGCNN, artificial start and end events were added to the traces. Therefore, the maximum trace length and thus prefix length is 11 instead of 9 (9+ 1 start event + 1 end event). Moreover, the first value for the prefix length in Table 42 is 3, since only prefixes with a minimum length of 3 were given as input to the BIG-DGCNN. (Chiorrini et al., 2022).

Table 42: CNN & BIG-DGCNN performance on the test partition of the total dataset per prefix length

NN	Prefix length	Number of prefixes	MAE (days)	95% CI
CNN	1	395925	9.198	[9.162,9.234]
	2	395925	9.171	[9.135,9.208]
	3	395925	9.042	[9.006,9.078]
	4	395925	8.745	[8.708,8.781]
	5	385919	6.858	[6.826,6.890]
	6	350951	4.856	[4.831,4.882]
	7	**241966**	**4.750**	**[4.720,4.780]**
	8	118987	5.172	[5.128,5.216]
	9	2996	7.211	[6.848,7.574]
BIG-DGCNN	3	396930	5.854	[5.822,5.885]
	4	396930	5.881	[5.850,5.912]
	5	396930	5.888	[5.857,5.919]
	6	396930	5.878	[5.846,5.909]
	7	386920	5.863	[5.831,5.894]
	8	351929	5.857	[5.824,5.891]
	9	242833	5.8455	[5.806,5.885]
	10	119480	5.8448	[5.789,5.901]
	11	3008	5.832	[5.473,6.190]

Figure 21 and 22 plot for both models the MAE for the different prefix lengths as well as the number of samples present for the prefix length.

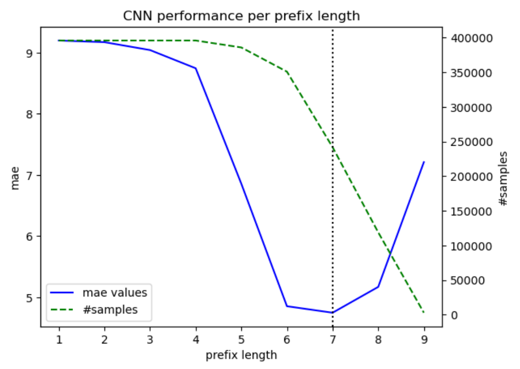


Figure 21: CNN performance for the total dataset (test partition) per prefix length

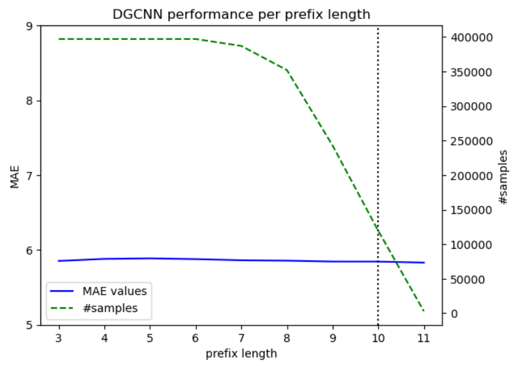


Figure 22: BIG-DGCNN performance for the total dataset (test partition) per prefix length

Both Table 42 and Figure 21 show that the best MAE value for the CNN is reached for prefixes with length 7. This value is a MAE of 4.750 with a 95%-CI of [4.720,4.780]. It means the CNN can make most accurate predictions when it has information of the last seven events that have occurred for a PO item. In terms of earliness the value would be prefix length 7, since this is the smallest prefix length where the model achieves an accuracy that is closest to the specified range of Company X (between 2 and 4 days). Interestingly, this best MAE value is substantially lower compared to the overall MAE (4.750 versus 7.619). Looking at the results in Table 42, it is evident that the overall MAE value is higher due to the higher MAE values for the shorter prefixes, especially the prefixes with length 1 until 5.
Figure 22 shows that for the BIG-DGCNN, the lowest MAE value is reached in the presence of a reasonable number of samples with prefix length 10. This value is a MAE of 5.8448 with a 95%-CI of [5.789,5.901]. However, when looking at the results for the BIG-DGCNN in Table 42, there is no prefix length for which the MAE value is clearly the best. For each prefix length, the 95%-CI seems to overlap with the 95%-CIs of all other prefix lengths. Therefore, there is no significant difference in performance in terms of MAE among the different prefix lengths for the BIG-DGCNN (Frost, 2019). This implies the BIG-DGCNN can make just as accurate predictions when it has information of the artificial start event and last two actual P2P events that have occurred for a PO item than

when it has information of the artificial start event and last nine actual P2P events for example. Especially, compared to the CNN the variation between MAE values for the different prefix lengths is minor. The BIG-DGCNN seems almost insensitive to the amount of information it gets about the proceeding of a trace. Since the MAE values are very similar for all prefix lengths the earliness would be the smallest prefix length, which is prefix length 3. Even though the earliness value is smaller for the BIG-DGCNN compared to the CNN, the corresponding MAE value is around 1 day more for the BIG-DGCNN.

Furthermore, the performance of both models is evaluated by investigating the MAE values for different type of cases based on the target values. Table 43 shows the MAE for the two NNs when considering only a specific type of case in the test set. Three types of cases are distinguished: too late (target$\in (0, 59]$), too early (target $\in (-69.001, 0)$), and exactly on-time (target $= 0$). Within the test set it is observed that for both NNs the MAE over all cases, 7.619 and 5.867 days for the CNN and BIG-DGCNN respectively, is increased by the absolute error values for the 'too late' cases. The MAE is especially high for those cases, with a value of 11.476 days for the CNN and 8.957 days for the BIG-DGCNN. It might be the case that the models have more difficulty predicting these cases, due to the fact that they are the minority group in the dataset. The model has then seen less samples for these cases during training. The 'too late' cases are 32.042 % of the total preprocessed dataset, whereas the 'too early' and 'exactly on-time' cases together account for 67.958 %.

Table 43: MAE values for different type of cases for the CNN and BIG-DGCNN trained on the total dataset (test partition)

NN	Case outcome	Target value range	Number of prefixes	MAE (days)	95% CI
	Too late	$(0, 59]$	894658	11.476	[11.447,11.505]
CNN	Too early	$(-69.001, 0)$	1619892	5.984	[5.972,5.997]
	Exactly on-time	0	169969	2.892	[2.867, 2.917]
	Too late	$(0, 59]$	898140	8.957	[8.929,8.985]
BIG-DGCNN	Too early	$(-69.001, 0)$	1623449	4.532	[4.520,4.543]
	Exactly on-time	0	170301	2.298	[2.275,2.321]

The MAE values for the different type of cases are lowest for the BIG-DGCNN. However, the difference in MAE value for the two models is only 0.594 days for the exactly on-time cases. The MAE 95% confidence intervals do not overlap for the two models and different type of cases. For example, for the 'too late' cases the BIG-DGCNN 95%-CI is [8.929,8.985], whereas the CNN 95%-CI is [11.447,11.505]. Therefore, the MAE values are significantly lower for the BIG-DGCNN (Frost, 2019).

The probability distributions of the absolute error values for the CNN and BIG-DGCNN are demonstrated by the histograms in Figures 23 and 24. Moreover, Figures 68 and 69 in Appendix B show how the absolute error values for the two NNs are distributed over the boxplot and shows quartiles, the median value, and outliers among others.

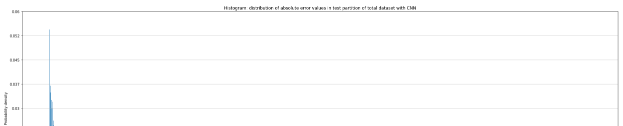


Figure 23: CNN distribution of absolute error values for the total dataset (test partition)

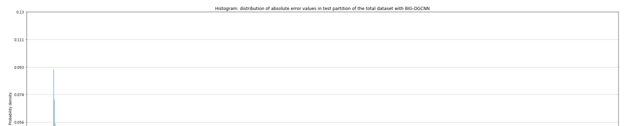


Figure 24: BIG-DGCNN distribution of absolute error values for the total dataset (test partition)

The two figures can be visually compared since the same bins have been used to create the two histograms. According to Figures 23 and 24, the BIG-DGCNN seems to have a higher percentage of extremely low MAE values. The peak in the plot around zero is at 0.090 for the BIG-DGCNN and at 0.054 for the CNN, indicating a higher proportion of low MAE values for the BIG-DGCNN. Moreover, the tail towards the higher MAE values is slightly thicker for the CNN.

The descriptive statistics regarding the absolute error values are summarized in Table 44.

Table 44: Descriptive statistics absolute error values for the CNN and BIG-DGCNN

NN	median	minimum	maximum	lower quartile	upper quartile	boxplot whiskers
CNN	3.673	5.000e-07	147.607	1.261	9.144	[5.000e-07, 20.970]
BIG-DGCNN	1.829	2.265e-06	159.611	0.470	6.525	[2.265e-06,15.607]

It can be observed that the median, lower quartile, and upper quartile have lower values for the BIG-DGCNN than for the CNN. For example, with the BIG-DGCNN for 75% of the prefixes the absolute prediction error is below 6.525 days, while with the CNN this is 9.144 days. This indicates that the BIG-DGCNN is performing better in terms of how the absolute prediction error values are distributed among the prefixes. However, the maximum absolute error is highest for the BIG-DGCNN, with a value of 159.611 days.

Lastly, the performance for both models is examined relatively to the target values. Therefore, the relation between the target values and absolute error values is depicted for the CNN and BIG-DGCNN respectively in Figure 25 and 26.

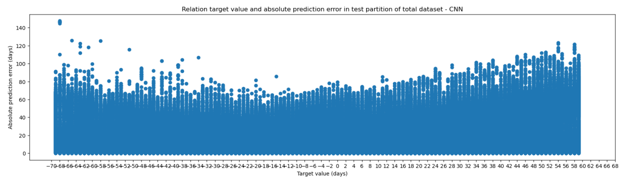


Figure 25: Relation target values and absolute error values for the CNN using the test partition of the total dataset

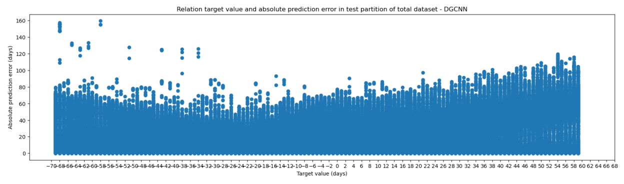


Figure 26: Relation target values and absolute error values for the BIG-DGCNN using the test partition of the total dataset

For both models somewhat a v-shape is visible in the figures plotting the relation between the absolute error values and target values, where the absolute errors seem to be higher for the more extreme target values, e.g. a target value of -60 or 50. When comparing the two figures there seem to be no clear differences. It could be argued that the extreme target values seem to affect the absolute error values more for the BIG-DGCNN, since there seem to be more extreme error values, also with higher values, when the target values are extremely negative. However, this is unexpected as a lower RMSE value was found for the BIG-DGCNN than for the CNN (11.637 compared to 13.110).

Finally, a conclusion needs to be drawn on which of these two models is performing best in terms of the evaluated metrics. However, it is also crucial that the model is suitable for implementation within Company X. This is then also taken into account within the final decision.
First of all, the MAE value is significantly lower for the BIG-DGCNN. Additionally, the SMAPE is lowest for the BIG-DGCNN. Besides, the RMSE value is lowest for the BIG-DGCNN implying the model is less sensitive to outliers. However, when looking at the scatter plots showing the relation between the target values and absolute error values, the CNN seems to be less sensitive for extreme target values. Furthermore, the MAE values seem to be lower for the too late cases with the BIG-DGCNN. It is especially important for Company X to predict those instances accurately, since actions need to be performed for those cases to prevent them from actually arriving late and to be able to improve the OTIF. On the other hand, the CNN gives the lowest MAE value of the two models when it has information of the past 7 events of a case (MAE=4.750 days). This value

comes closest to what Company X is striving for in terms of prediction error, which is 2-4 days. However, the BIG-DGCNN has already quite a good MAE performance when having information of only the past two events of a case (prefix length 3), with a value of 5.854 days. Moreover, the MAE value stays rather similar when more information about the proceeding of a case is available, unlike for the CNN. This shows that the BIG-DGCNN is more stable and therefore more reliable than the CNN. However, since the MAE values drop for the CNN when information about more events is available, it reaches lower MAE values than the BIG-DGCNN which as well get closer to the desired values by Company X. But, these MAE values also increase again when the prefix length becomes bigger than 7.
Furthermore, the training time is much shorter for the CNN, with a difference of almost 2 hours per epoch. The model could process the prefixes in the total dataset using batch size 64 within 1.041 hours/epoch during model training, while this was 2.918 hours/epoch with the BIG-DGCNN using an even higher batch size. Furthermore, predictions were generated faster by the CNN compared to the BIG-DGCNN on the test partition of subsample $k = 6$ (0.0629 versus 0.317 hours). The results of subsample $k = 6$ are used for a fair comparison of prediction time since there a batch size of 1 was used for the test sets for both models. To obtain the BIG-DGCNN results on the test partition for the total dataset, the batch size had to be set to half the size of the test partition to prevent getting a memory error. Therefore, it can be said that the CNN is more efficient in terms of time and memory when generating the predictions. This is especially important when deploying the model in the Celonis environment of Company X.

Although, the BIG-DGCNN shows a more promising performance in terms of prediction accuracy, the CNN is chosen to be most suitable model to implement for Company X. This choice is mainly based on the training and prediction efficiency. The CNN trains faster, generates predictions faster and consumes less memory. Especially the memory consumption is important, since only 4GB is available in the Celonis ML workbench to allocate to the execution of the script to generate the predictions. Moreover, the necessary preprocessing process to transform the cases into the graph inputs for the BIG-DGCNN is extremely time consuming, which makes the model less appropriate for implementation. With subsample $k = 6$, containing 241986 cases and 1654756 prefixes, it took approximately 5 hours to perform the BIG-algorithm to create the instance graphs for the cases, do the feature engineering for the cases, and create the graph objects for all prefixes. Company X has around 150,000 cases (PO items) for which the predictions need to be made each day. Although only one prediction has to be generated for each case (and not for every prefix of the case like when training the model), the preprocessing is still expected to take a couple of hours. For example, the BIG-algorithm and feature engineering are performed on case-level and these steps took already around 2.5 hours for subsample $k = 6$. Furthermore, the BIG-DGCNN requires adding an artificial start and end event to the traces in order for the BIG-algorithm to find the correct alignments between the traces and the process model (also artificial start and end events are present in the process model). The artificial events cannot be added to the traces with the earlier used ProM plug-in, but this addition process has to be performed in the Python environment in Celonis. This is expected to be a cumbersome and time-consuming process within Python.

In the following subsection, it is investigated whether the random forest approach can even further improve predictive performance.

Random Forest Approach
This subsection elaborates on the experiments conducted for the Random Forest approach by analyzing the results for the different RF configurations as given in Table 56 in Appendix B, as well as by examining the NN performance for the distinct types of cases (see Tables 50 - 53).

One of the hypothesized advantages of the RF approach was that the NNs only needed to make predictions for specific case types, thus could be trained on smaller ranges of target values, which would decrease the prediction error. Like explained in Section 5.1.2, the RF models would allow the NNs to be trained either only on the 'too late' cases (target value $\in (0, 59]$) or the 'normal cases'

(target value $\in (-20, 20)$). In order to validate the above assumption, the results are analyzed for the five NNs when only trained on the 'too late' and 'normal' cases.
First, the results for the 'too late' cases are analyzed. It is evident that MLP performance in terms of validation MAE is not improving greatly by running the model only on the 'too late' cases. The MAE drops from 10.217 to 9.875 days with subsample $k = 6$, which is only a difference of 0.342 days on average. A more clear improvement in validation MAE is found for the GCN where the value drops from 13.096 to 11.104 with subsample $k = 6$, giving a difference of 1.992 days on average. It was expected that performance would improve more due to the removal of cases with a target value $\in (-69.001, 0]$, being 66.334% of all cases, and thus decreasing. Since a positive relation between extreme target values and extreme MAE values was found in scatter plots for both NN types, a smaller MAE was expected with less extreme values being present. However, within the 'too late' cases still extreme values are present, like the maximum of 59 days too late. This might still lead to a high average for the absolute error. Training time has decreased from 0.497 to 0.167 hours per epoch for the MLP and from 0.155 to 0.071 hours per epoch for the GCN with subsample $k = 6$. This seems reasonable, because there are less cases present in the subsample after dropping the very early and on-time cases.
Next, the results for the 'normal' cases are analyzed. The results are summarized in Table 45. For all of the NNs, the performance in terms of validation MAE is visibly improving when running the model only on the 'normal' cases. A possible reason could be the fact that when training the models only on the (-20,20) range, a smaller range of values needs to be predicted which makes it easier for the model to predict correctly. Moreover, no extreme target values are present in the dataset anymore. This might reduce the number of times a large absolute error occurs, since it was observed that higher absolute errors occurred for the prefixes with more extreme target values to predict. Especially, a great improvement is observed for the GCN and LSTM, where the MAE values have decreased with 7.043 and 7.580 days respectively. Probably, these NN types are more sensitive to extreme target values in the dataset than the other models. The BIG-DGCNN and CNN have the smallest decrease in validation MAE with respectively 3.352 and 4.251 days. Therefore, extreme target values seem to have a smaller effect on performance for these models.

Table 45: Performance comparison validation set for only 'normal' cases (days too late $\in (-20, 20)$) on subsample k=6

Model	Specific parameter values	MAE (days)		training time (hours/epoch)	
		All cases	'normal' cases	All cases	'normal' cases
MLP	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001.	10.217	5.024	0.497	0.417
GCN	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001.	13.096	6.053	0.155	0.130
LSTM	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001.	13.642	6.062	0.0701	0.0506
CNN	own features, 128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001.	8.599	4.348	0.166	0.118
BIG-DGCNN	time features of Chiorrini et al. (2022) and own non-time features, learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5.	6.627	3.275	2.352	1.448

For the purpose of being able to actually train the models on these smaller ranges, the RF performance should be good enough. Otherwise, a big error could be created by making a prediction based on a previous prediction, where the quality of the first prediction is already insufficient. Therefore, the performance of the different RF configurations is analyzed based upon the results for the various conducted experiments, see Table 56 in Appendix B. The evaluation metrics that are used to analyze the models are explained in Section 5.1.3.

First of all, the RF with the three classes, each containing an equal proportion of the data, is evaluated. The F1-macro score is fairly high with a value of 0.685. Moreover, the precision, recall,

and F1-scores are relatively similar for all three classes. This is probably because of the equal proportion of data for each of the three classes. The highest performance values are found for class zero (precision=0.708, recall=0.818, and F1=0.759). Therefore, the model is is best at predicting the cases which are delivered much before the contractual delivery date.
The RF with two classes to predict has a F1-macro score of 0.664 which is as well relatively high. Especially, class zero ($(-69.001, 0]$) is well predicted, having the highest score for precision, recall, and F1. This could have been expected, since the highest percentage of data is within this class (67%). The training time is much longer than for the equal bins approach (39.441 versus 3.612 hours), although the same number of trials are performed for hyper-parameter optimization. This is owed to the fact that 5-fold cross-validation is implemented.
When inspecting the RF smaller bins: The F1-macro score drops to 0.512. This is because the recall and F1 get really bad for class 2. Class 2 is also only 14% of the dataset so that influences the results.
Next, the results are assessed for the RFs that had three classed to predict that were based on domain-knowledge. It is discovered that the results are almost equal in the following cases: (1) when the objective is maximizing the mean validation accuracy, (2) the objective is maximizing the mean validation F1-micro score, (3) or maximizing the composite loss being the product of mean validation accuracy, mean validation precision, and mean validation recall. This is in line with what was found in literature: " Micoaverage precision, recall, F1 and accuracy are all equal for cases in which every instance must be classified into one (and only one) class" (Shmueli, 2019). Moreover, class 2, which refers to the very late cases (target $\in [20, 59]$), is very poorly recognized by the RF in all three configurations. Hence, the recall value is very low: (1) 0.058, (2) 0.057, and (3) 0.053 respectively. This insufficient result for class 2 is reasonable since it is the minority class and represents only 14% of the dataset. On the other hand, the precision values are reasonable: (1) 0.607, (2) 0.622, and (3) 0.639. This means that the proportion of correct predictions for class 2 is fairly well. Another observation is that the accuracy in all three RF configurations (0.729, 0.730, 0.746) is almost equal to the proportion of the majority class in the dataset, which is 67%. This means that the models do not have much predictive power, and are mostly good at predicting the majority class.
According to Stack Exchange Inc (2020) macro-averaging is suggested to be more useful when having imbalanced data, since it weights each class equally in calculating an average score over the classes. Therefore, the RF results are expected to be better, especially for the minority class (class 2), when the mean validation F1-macro score is used as objective for the hyper-parameter optimization objective. However, this is not entirely confirmed. For example, comparing the values for the evaluation metrics only for class 2 between the RF with F1-micro as objective and RF with F1-macro as objective, it is seen that precision is declining (0.622 versus 0.475), and recall and precision are improving to a minor extent (precision: 0.057 versus 0.0.076, F1-score: 0.105 versus 0.132). When using stratified cross-validation, so keeping the distribution of the target the same in each fold, as well as some kind of data balancing, the results for the minority class improve. However, it cannot be said whether one of the two constructs has more influence than the other because they have not been tested in isolation. It is observed that precision has declined for class 2. For instance, when comparing precision between the RF with F1-macro as objective and normal cross-validation and the RF with random oversampling and stratified cross-validation, the value decreased from 0.475 to 0.377. However, recall improved extremely. For example, when comparing the recall for the same RFs it is observed that the value went up from 0.076 to 0.294. For other RFs using data balancing and stratified cross-validation, the value went even up to 0.444 or 0.539. This means that this minority class is much better recognized by the models.
The subsequent paragraph elaborates on the final comparison between the different RF configurations and explains whether the RF approach is suitable for implementation within Company X or not.

Final comparison of Random Forest configurations
When making the decision on which RF configuration had the best performance, only the ones where the classes to predict were based on domain-knowledge are considered. The reason is that it

was seen that NN performance is not improving greatly when training the models only on the ‘too late’ cases, but is substantially improving when trained only on the ‘normal’ cases. The ‘normal’ cases is a class that is being predicted only by the RFs with domain-knowledge based bins. The RF configurations with the highest F1-macro score are the RF with over- and undersampling resulting in an equal percentage of data for the three classes, the RF with oversampling the minority classes to 60% of the majority class, the RF with balanced subsample class weight parameter, and the balanced RF. Their scores are 0.562, 0.582, 0.576, and 0.558 respectively. To decide which of these models is performing best, the percentages for the incorrect predictions of category 3, 5, and 6 are checked (see Section 5.1.3). The results are shown in Table 46. It shows for example that with the RF having a balanced subsample class weight, a proportion of 0.182 of all PO items that would actually be delivered between 19 days early and 19 days late is predicted to be delivered extremely early.

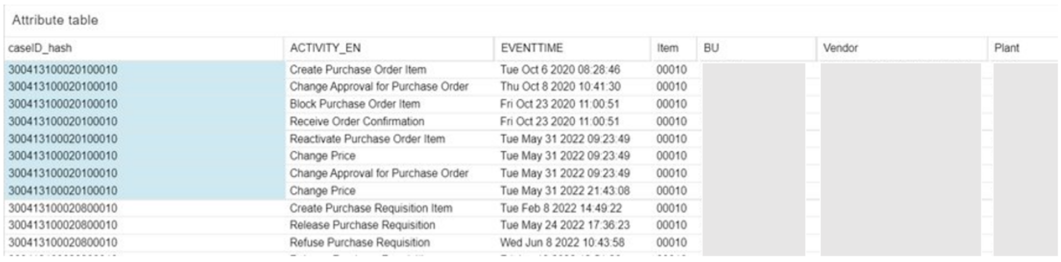
Table 46: Deeper performance evaluation best RF models

RF type	Category 3	Category 6	Category 5
RF with over- and undersampling	0.180	0.272	0.220
RF with oversampling the minority classes	0.162	0.493	0.213
RF with balanced subsample class weight parameter	0.182	0.334	0.231
The Balanced RF	0.179	0.238	0.223

Overall, the balanced RF has the lowest proportion of incorrect predictions of category 3 and 6 compared to the other RFs in Table 46. Moreover, the proportion for category 5 is the third lowest. Therefore, this RF is chosen to be the best performing. Although, the proportions are lowest for the balanced RF, the values are not good enough to implement this Random Forest approach. Firstly, 46.1% of the predictions generated for the cases that actually are delivered between 20 and 59 days later than the contractual date, are incorrect and are either of category 5 or 6 and thus highly unwanted by Company X. Especially, the 17.9% of items that will be delivered very late but are predicted to be delivered very early, are problematic. If the RF approach would be implemented with this model, these cases are not predicted to arrive late and thus no action is taken before the items are actually already overdue. Moreover, the precision for the too late class is only 0.279 which is extremely low. If the RF approach would be implemented with the balanced RF, 89.1% of the items that are predicted to be arriving in the normal range, are predicted correctly. Then a prediction is produced for the difference in days between actual and contractual delivery date for these items, where the prediction is on average 4.348 days off. However, 10.9% of these cases going through the best NN are not normal cases giving an actual bigger error for these cases. Without the implementation of the RF, the best NN is trained on all cases and would still have an MAE of only 0.9 days more (MAE = 5.248 days) for the ‘normal’ cases. Therefore, it is decided that it is better for Company X to just implement the best NN and train it on all cases. The final implementation of the best NN is further described in the next section.

5.2.4 Deployment

Like explained in Section 5.1.4, the best predictive model could be implemented within the Celonis environment of Company X by adding the trained model as well as a Python script performing the predictions to the integrated ML workbench. Therefore, the CNN is added to the ML workbench, which was trained on the total dataset. Next, this Python script is used to load the data table with all the necessary input variables for the new prefixes. An example of part of the Celonis input data table is given in Figure 27. Note that data for the BU, vendor, and plant are hidden due to confidentiality.

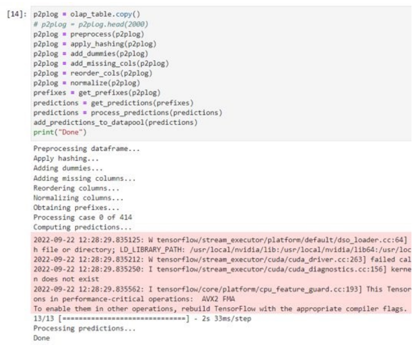


Attribute table

caseID_hash	ACTIVITY_EN	EVENTTIME	Item	BU	Vendor	Plant
300413100020100010	Create Purchase Order Item	Tue Oct 6 2020 08:28:46	00010	[redacted]	[redacted]	[redacted]
300413100020100010	Change Approval for Purchase Order	Thu Oct 8 2020 10:41:30	00010	[redacted]	[redacted]	[redacted]
300413100020100010	Block Purchase Order Item	Fri Oct 23 2020 11:00:51	00010	[redacted]	[redacted]	[redacted]
300413100020100010	Receive Order Confirmation	Fri Oct 23 2020 11:00:51	00010	[redacted]	[redacted]	[redacted]
300413100020100010	Reactivate Purchase Order Item	Tue May 31 2022 09:23:49	00010	[redacted]	[redacted]	[redacted]
300413100020100010	Change Price	Tue May 31 2022 09:23:49	00010	[redacted]	[redacted]	[redacted]
300413100020100010	Change Approval for Purchase Order	Tue May 31 2022 09:23:49	00010	[redacted]	[redacted]	[redacted]
300413100020100010	Change Price	Tue May 31 2022 21:43:08	00010	[redacted]	[redacted]	[redacted]
300413100020800010	Create Purchase Requisition Item	Tue Feb 8 2022 14:49:22	00010	[redacted]	[redacted]	[redacted]
300413100020800010	Release Purchase Requisition	Tue May 24 2022 17:36:23	00010	[redacted]	[redacted]	[redacted]
300413100020800010	Refuse Purchase Requisition	Wed Jun 8 2022 10:43:58	00010	[redacted]	[redacted]	[redacted]

Figure 27: Part of the input data table within Celonis

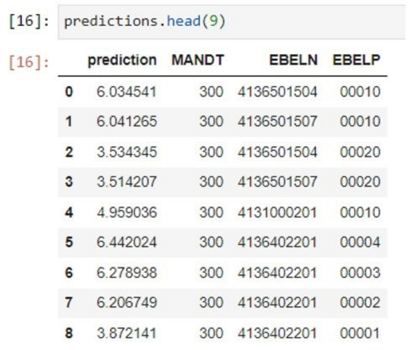
Subsequently, the Python script transforms this input data into the correct prefix format by executing all the preprocessing functions like explained by the pseudocode in Algorithm 2. This process within the ML workbench is shown in Figure 28.



```
[14]: p2plog = olap_table.copy()
      # p2plog = p2plog.head(2000)
      p2plog = preprocess(p2plog)
      p2plog = apply_hashing(p2plog)
      p2plog = add_dummies(p2plog)
      p2plog = add_missing_cols(p2plog)
      p2plog = reorder_cols(p2plog)
      p2plog = normalize(p2plog)
      prefixes = get_prefixes(p2plog)
      predictions = get_predictions(prefixes)
      predictions = process_predictions(predictions)
      add_predictions_to_datapool(predictions)
      print("Done")

      Preprocessing dataframe...
      Apply hashing...
      Adding dummies...
      Adding missing columns...
      Reordering columns...
      Normalizing columns...
      Obtaining prefixes...
      Processing case 0 of 414
      Computing predictions...
      2022-09-22 12:28:29.835125: W tensorflow/stream_executor/platform/default/dso_loader.cc:64]
      h file or directory; LD_LIBRARY_PATH: /usr/local/nvidia/lib:/usr/local/nvidia/lib64:/usr/loc
      2022-09-22 12:28:29.835212: W tensorflow/stream_executor/cuda/cuda_driver.cc:263] failed cal
      2022-09-22 12:28:29.835250: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kerne
      n does not exist
      2022-09-22 12:28:29.835562: I tensorflow/core/platform/cpu_feature_guard.cc:193] This Tensor
      ons in performance-critical operations:  AVX2 FMA
      To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
      13/13 [==============================] - 2s 33ms/step
      Processing predictions...
      Done
```

Figure 28: Python script within the ML workbench of Celonis



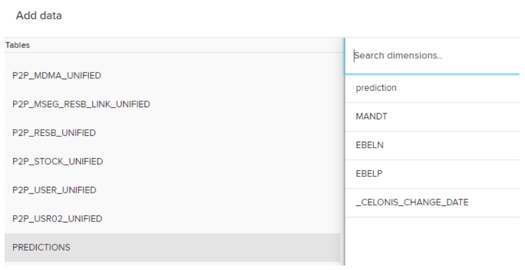
```
[16]: predictions.head(9)
```

[16]:

	prediction	MANDT	EBELN	EBELP
0	6.034541	300	4136501504	00010
1	6.041265	300	4136501507	00010
2	3.534345	300	4136501504	00020
3	3.514207	300	4136501507	00020
4	4.959036	300	4131000201	00010
5	6.442024	300	4136402201	00004
6	6.278938	300	4136402201	00003
7	6.206749	300	4136402201	00002
8	3.872141	300	4136402201	00001

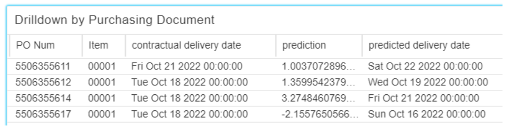
Figure 29: The resulting predictions within the ML workbench of Celonis

The demonstrated *get_predictions* and *process_predictions* functions in Figure 28 include the process of generating the predictions through loading the CNN and applying it to the preprocessed input. Besides, the process of tracing back the predictions to the corresponding PO items. The result of executing the entire Python script is shown in Figure 29. These are the predicted number of days between the actual delivery date and the contractual delivery date for each of the PO items. These PO items are identified by their case ID which is a combination of client (MANDT), purchasing document (EBELN) and item number of purchasing document (EBELP). After these results are produced, the table like shown in Figure 29 is added to the datapool. Hence, the predictions can be retrieved within the expedite dashboard of Company X and added to their analysis, like shown in Figure 30. In order to get the final predicted delivery date, the predicted number of days between the actual delivery date and the contractual delivery date is added to the contractual delivery date within the analysis in the expedite view. A snapshot of part of this analysis for Company X is shown in Figure 31.


Add data

Tables	Search dimensions..
P2P_MDMA_UNIFIED	prediction
P2P_MSEG_RESB_LINK_UNIFIED	MANDT
P2P_RESB_UNIFIED	EBELN
P2P_STOCK_UNIFIED	EBELP
P2P_USER_UNIFIED	_CELONIS_CHANGE_DATE
P2P_USR02_UNIFIED	
PREDICTIONS	

Figure 30: Prediction data table can be added to Celonis analysis


Drilldown by Purchasing Document

PO Num	Item	contractual delivery date	prediction	predicted delivery date
5506355611	00001	Fri Oct 21 2022 00:00:00	1.0037072896...	Sat Oct 22 2022 00:00:00
5506355612	00001	Tue Oct 18 2022 00:00:00	1.3599542379...	Wed Oct 19 2022 00:00:00
5506355614	00001	Tue Oct 18 2022 00:00:00	3.2748460769...	Fri Oct 21 2022 00:00:00
5506355617	00001	Tue Oct 18 2022 00:00:00	-2.1557650566...	Sun Oct 16 2022 00:00:00

Figure 31: The final predicted delivery dates within the Celonis analysis of Company X

6 Prescriptive Process Monitoring

This chapter elaborates on all the steps specifically performed for the prescriptive part of this thesis (see Figure 3). This part of the thesis focuses on the creation of a prescriptive model that recommends which purchased items from suppliers need to be prioritized for expediting for the day, to maximally impact the OTIF rate. First, all steps are explained that have been performed regarding data preparation, method selection and implementation, evaluation, and deployment. Thereafter, the results are discussed with respect to model implementation, evaluation, and deployment.

6.1 Prescriptive Process Monitoring Steps

Within this section the method is explained for the performed steps regarding the prescriptive part of this thesis.

6.1.1 Data preparation

After the business understanding and data understanding phase, the *data preparation* phase was performed starting with the selection of the required data by applying the earlier defined filters in Table 11. Next, the problems regarding data quality were solved by correcting attribute values, removing invalid cases, or substituting missing values. Finally, since the prescriptive model was created within Celonis, the data did not have to be exported from Celonis but could be directly used within Celonis for the evaluation and implementation of the model.

6.1.2 Method Selection and Implementation

Like explained in Section 3.4, Company X has an expedite dashboard which shows the list of prioritized PO items for expediting. The dashboard displays all PO items that are not yet delivered and their corresponding details like material criticality, contractual delivery date, vendor, OTIF of last year, etc. Their original prescriptive model works such that within this dashboard only the items are being selected that are already overdue. Subsequently, the expedite score is being calculated for those items using the weighted scoring model as explained in Section 3.4. Hence, the items with the highest expedite score will get the highest priority for expediting.

As mentioned in Section 3.4, the newly developed prescriptive model was based upon the existing expedite dashboard and weighted scoring model of Company X. In order to enable Company X to act proactively regarding purchased items and to improve their decision making with respect to the expediting process, several changes were made to existing expedite framework. This finally resulted in the newly developed prescriptive model. First of all, the range of items being expedited was changed. An extra filter was added to the dashboard such that not only the already overdue items could be selected, but also the items that will become overdue within the upcoming 14 days (when the planned delivery date is passed) and are predicted to be more than 1 day late. The 14 days and 1 day are numbers determined by the supply chain manger of Company X. Hence, the expedite score would be calculated for a larger number of PO items, including items that are not yet overdue. Subsequently, for all these items (overdue and soon to be overdue) a new expedite score was calculated where the factor 'days overdue' was changed. The factor 'days overdue' was changed such that it took the predicted delivery date into account as produced by the predictive model developed in the previous part of this thesis. The score for 'days overdue' was changed into the following: the **maximum** of the date of Today minus the planned delivery date, and the predicted delivery date minus the planned delivery date. The value for 'days overdue' would then either be the number of days the item is currently overdue OR the number of days the item is predicted to be overdue. Then, for each 2 days the PO item is overdue or predicted to be overdue, the score would increase with 1 point, to a maximum of 10 points. By changing this factor in the original framework, the expedite score took different values for the PO items which resulted in a new and different priority list for expediting. The effects of the new prescriptive model on Company X's operations were evaluated as explained in the next section.

6.1.3 Evaluation

In order to asses how the prescriptive model would bring value to Company X, the model needed to be evaluated. Therefore, a historical analysis was performed to evaluate the model on past expedited items, as well as a future analysis to evaluate the impact of the model on the expediting of the most critical items. In the following, these two analyses are explained in more detail.

1. *Historical analysis*: This analysis aimed at investigating the proportion of past expedited items that could have been acted on proactively in case the prescriptive model would have been implemented at that time. This was achieved by checking the percentage of past expedited items for which it was predicted, by the implemented CNN, that they would be delivered later than the contractual delivery date.
   Hence, a list was created of PO items that were expedited in the past by the expediting team of Company X. This means these items were overdue (not delivered while the contractual delivery date was already passed) in the past and therefore expedited. To clarify, it could be the case that these items are currently still not delivered, since the supplier indicated during expediting that the delivery date was moved forward a couple of months. This list could be created since the expediting records, i.e. information about the supplier contact moments for the PO items, are stored in Celonis. Only expedite records for PO items created later than January 2022 were included in the analysis. This is because only those records were indicated to be reliable by the process mining expert, since the expedite team only really started recording their expedite actions from that point in time. Next, for these items the prediction for the number of days between the actual and contractual delivery date was retrieved, to see whether the items were predicted to be delivered late or not. These predictions could be retrieved since the CNN was deployed in Celonis (see Section 5.2.4). Subsequently a percentage could be retrieved of the PO items for which the CNN was capable of predicting that they would be delivered late. This percentage gives an indication of the impact of the prescriptive model. That is, because it shows the number of past expedited items that could have been expedited before they were actually overdue because of the predictions, since the prescriptive model allows for including items in the priority list for expedited which are predicted to arrive late. These items could then maybe have been prevented from being delivered late if the model was used. Furthermore, the order value of these items was assessed to also give an indication of financial risk that could be reduced if these items are possibly delivered more on-time.

2. *Future analysis*: Next, the future analysis aimed at investigating the effect of the newly developed prescriptive model on the priority for expediting for the most critical purchased items. This is interesting to examine as especially late delivered critical materials can disrupt Company X's maintenance processes. Therefore, the percentage was examined of the most critical PO items which were not yet overdue but would become overdue soon and were predicted to be delivered more than 1 day late. This percentage indicates the number of very critical PO items that can now be expedited before they are actually overdue, since the new prescriptive model includes these items in the expediting priority list, and thus be prevented from being delivered late. This would show the extent to which Company X is allowed to act more proactively. Therefore, after the prescriptive model was deployed in the expedite dashboard of Company X, a list of PO items was created which had the highest material criticality (C1 or 1) and which were not yet overdue but would become overdue in the upcoming 14 days as well as were predicted to be more than 1 day late. Subsequently, the percentage was checked of the not yet delivered PO items that were included in this list. Furthermore, it was investigated how much order value was involved for those items, to give an indication of financial risk that could be reduced if these items are possibly delivered more on-time.
   To further asses the impact of the new way of prioritizing PO items on the OTIF rate, it was examined how much sooner the most critical items would be expedited with the new prescriptive model. This was explored by first comparing the new expedite score with the as-is expedite score for the items in the list. Moreover, by comparing the position of items in the priority list when sorted on either the new expedite score or as-is expedite score.

6.1.4 Deployment

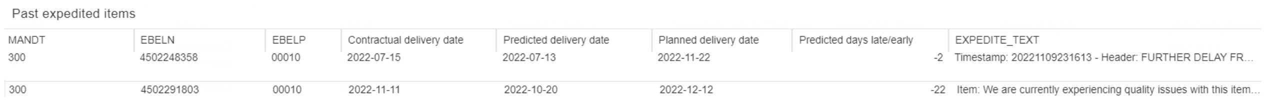
The prescriptive model was finally implemented within the Celonis environment of Company X. Since the predictive model was implemented within Celonis as well (see 5.1.4), the values for the predicted delivery date for items could be retrieved within the existing expedite dashboard. The implementation of the prescriptive model was tested within this existing expedite dashboard in Celonis. Firstly, the filter ‘orderline overdue’ was changed in the expedite dashboard such that it became possible to select all PO items that are becoming overdue in the upcoming 14 days and are predicted to be more than 1 day late. Hence, these items could be added to the expedite priority list. Moreover, a new formula for the expedite score was added within the expedite dashboard including the updated computation for factor ‘days overdue’. A detailed explanation of the deployment is provided in Section 6.2.2 within this chapter.

6.2 Prescriptive Process Monitoring Results

The second part of this chapter focuses on the results for the prescriptive process monitoring part of the thesis. First, the results for the historical of the prescriptive model, as described in Section 6.1.3, is explained. Then, in order to do the future analysis, the prescriptive model first needs to be deployed within the expedite dashboard of Company X in Celonis. Therefore, the deployment of the prescriptive model is elaborated after the historical analysis, and followed by the future analysis. This section finishes with a discussion on the reliability of the new expedite score within the prescriptive model.

6.2.1 Historical Analysis

After applying the process filters to select the cases for this historical analysis (see Section 3.6), 147 PO items are included. This means that 147 PO items which were created in the period of February 2022 until September 2022 and were expedited in the past. It is found that for 113 of these items the implemented CNN predicted that they would be delivered later than the contractual delivery date. This is for 76.870% of the items. The total order value of these 113 items is over 1 Million dollars. Hence, it can be concluded that the model could have predicted a fairly high percentage of items to be too late in the past. This means if Company X would have implemented the predictive model already in these past 9 months (February-September), these items could have been expedited before the items were overdue instead of at the moment the items were already overdue. Although 113 items might not sound as a lot, the order value is absolutely high, implying a real impact for Company X.
On the other hand, there are still 34 items (23.130%) which were expedited in the past, but for which the CNN predicted the items to arrive exactly on-time or even before the contractual delivery date. Two examples of such items are shown in Figure 32. For example, the first item in the figure was predicted to be delivered 2 days earlier than the contractual delivery date, but was eventually overdue and expedited, where the supplier informed that the item would only be delivered 4 months later. Although these incorrect predictions occur for the minority of the past expedited items, the CNN still seems to occasionally make too positive predictions for items, which were in reality much delayed.


Past expedited items

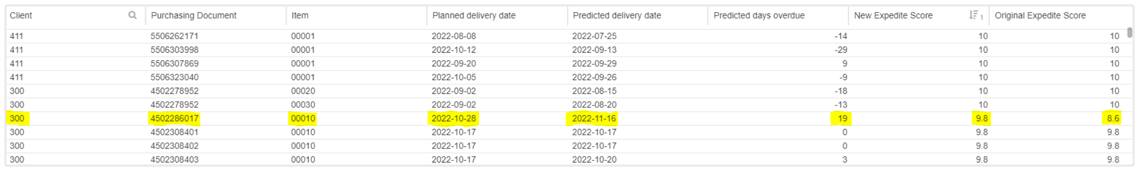
MANDT	EBELN	EBELP	Contractual delivery date	Predicted delivery date	Planned delivery date	Predicted days late/early	EXPEDITE_TEXT
300	4502248358	00010	2022-07-15	2022-07-13	2022-11-22	-2	Timestamp: 20221109231613 - Header: FURTHER DELAY FR...
300	4502291803	00010	2022-11-11	2022-10-20	2022-12-12	-22	Item: We are currently experiencing quality issues with this item...

Figure 32: Incorrectly predicted PO items

Since this analysis includes only the items which were expedited in the past and thus were overdue at some point, it could not be examined within this analysis whether the CNN predicted items to be delivered too late while in reality they were delivered on-time. However, since this is an interesting point to explore, a selection was made of the PO items which were not expedited and for which the delivery was completed. However, none of these items seemed to have a predicted delivery date due to the CNN being implemented later than when the items were delivered. Therefore, it could still not be explored whether there were items incorrectly predicted to be delivered late by the implemented CNN.

6.2.2 Deployment

When implementing the prescriptive model within the existing expedite dashboard of Company X, the following elements were changed. First, the filter ‘order line overdue’ was changed in the expedite dashboard. The options for this filter were first only ‘yes’ and ‘no’ indicating whether to include overdue orders or not overdue orders in the expedite priority list. Another option was added to this filter, namely ‘Predicted Overdue’, to select all PO items that are becoming overdue in the upcoming 14 days and are predicted to arrive more than 1 day later than the contractual delivery date. Hence, when checking the box for ‘Predicted Overdue’ these items could be added to the expedite priority list. Behind this filter a KPI is created that checks for each PO item whether it is overdue, not yet overdue, or predicted to be soon overdue, by looking at the date of “Today”, the planned delivery date, and predicted number of days the item is expected to be late compared to the contractual delivery date. Furthermore, a new filter is created, ‘Include prediction’, to indicate whether the expedite score needs to be changed to the new score or not. If the box ‘Predicted Overdue’ is checked, the expedite score is changed to the new score including the predicted delivery date for PO items. To use this new expedite score within the expedite dashboard, a new KPI is added to the dashboard containing the new formula to calculate the score as described in Section 6.1.2. Subsequently, this expedite score can be added to the list of selected PO items such that the items can be sorted on the new expedite score, and thus prioritized for expediting.
Figure 33 displays the expedite priority list (created at the 4th of November 2022), when selecting the PO items using the process filters defined for the implementation of the prescriptive model (see Section 4.2 and sorting the PO items on the new expedite score. The column ‘predicted days overdue’ is the value for (predicted delivery date - planned delivery date) and thus not directly the prediction from the CNN.



Client	Purchasing Document	Item	Planned delivery date	Predicted delivery date	Predicted days overdue	New Expedite Score	Original Expedite Score
411	5506262171	00001	2022-08-08	2022-07-25	-14	10	10
411	5506303998	00001	2022-10-12	2022-09-13	-29	10	10
411	5506307869	00001	2022-09-20	2022-09-29	9	10	10
411	5506323040	00001	2022-10-05	2022-09-26	-9	10	10
300	4502278952	00020	2022-09-02	2022-08-15	-18	10	10
300	4502278952	00030	2022-09-02	2022-08-20	-13	10	10
300	4502286017	00010	2022-10-28	2022-11-16	19	9.8	8.6
300	4502308401	00010	2022-10-17	2022-10-17	0	9.8	9.8
300	4502308402	00010	2022-10-17	2022-10-17	0	9.8	9.8
300	4502308403	00010	2022-10-17	2022-10-20	3	9.8	9.8

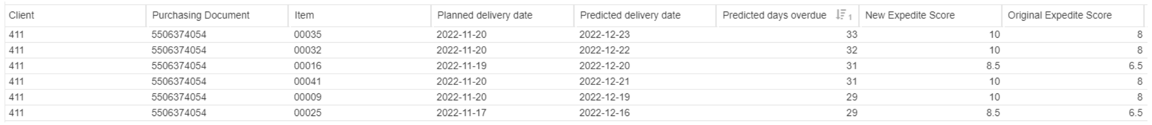
Figure 33: Expediting Priority list

It is noticed that one of the most urgent items for expediting is a PO item that is not yet overdue but predicted to arrive very late. Due to the taking the predicted delivery date into account within the new expedite score this item appears higher on the priority list than when taking the predicted delivery date not into account. If the original expedite score was used for prioritizing, the same item would have been shifted approximately 150 PO items down on the priority list.

6.2.3 Future Analysis

Only the items are evaluated for which the planned delivery date is within 14 days, meaning the item will be overdue within 14 days if it is not delivered by then. Besides, the items are expected to be more than 1 day late. The analysis is performed at the 15th of November 2022. This is an important note since the items included in the list are dependent upon the day of analysis since the items that are due within 14 days from the moment of analysis are included in the evaluation.

It is found that 207 out of the 2043 PO items, with the highest material criticality and which are not yet overdue, are becoming overdue within the upcoming 14 days and are predicted to be delivered more than 1 day late. This accounts for 10.132% of the most critical PO items that are not yet delivered and overdue. The order value of these items together is over a couple of Million dollars. It is noted that the new expedite score is higher for the selected items compared to the original expedite score due to changing the factor 'days overdue', however, only up to a maximum of 2 points difference. This is because the maximum score for this factor is 10 points and is weighted by 20%. For example, Figure 34 shows several PO items that are predicted to become overdue in the upcoming 14 days (from 15-11-2022) and more than 1 days late. It is visible that all the items are not too late yet, however, since they are predicted to be delivered much later than the planned delivery date ($\geq$ 20 days), their expedite score has gone up by the maximum of 2 points. These items will get a much higher priority for expediting.



Client	Purchasing Document	Item	Planned delivery date	Predicted delivery date	Predicted days overdue	New Expedite Score	Original Expedite Score
411	5506374054	00035	2022-11-20	2022-12-23	33	10	8
411	5506374054	00032	2022-11-20	2022-12-22	32	10	8
411	5506374054	00016	2022-11-19	2022-12-20	31	8.5	6.5
411	5506374054	00041	2022-11-20	2022-12-21	31	10	8
411	5506374054	00009	2022-11-20	2022-12-19	29	10	8
411	5506374054	00025	2022-11-17	2022-12-16	29	8.5	6.5

Figure 34: Prescriptive model: Future analysis within Celonis

Next, it is checked whether a higher expedite score, due to the item predicted to be delivered late, also really leads to a higher priority for expediting. Therefore, the position within the priority list has been compared for several items, between the list sorted on the new expedite score and the list sorted on the original expedite score. For this analysis, all PO items with the highest criticality are included that need to be expedited, so also the items that are already overdue. Table 47 shows the results for multiple PO items.

Table 47: Priority comparison for the most critical PO items

Client	Purchasing document	Item	New position in priority list	Original position in priority list	New expedite score	Original expedite score	Planned delivery date	Days overdue	Predicted days overdue
411	5506350380	00002	76	113	8.5	7.1	04-11-2022	11	21
411	5506350380	00003	77	114	10	9	04-11-2022	11	21
411	5506374054	00005	79	391	10	8	20-11-2022	0	28
411	5506376787	00001	87	399	10	8	25-11-2022	0	25
411	5506366797	00001	121	390	9.2	8	18-11-2022	0	13
411	5506350380	00001	120	112	9.2	9	04-11-2022	11	13

Interestingly, there is one item listed in Table 47 for which the priority did not increase while the expedite score increased. The underlying reason is the fact that with the original calculation of the expedite score, there are less items with an extremely high expedite score. By changing the calculation for the expedite score such that it takes the predicted delivery date into account, there are more items which have gotten a higher priority for expediting. Hence, a higher expedite score does not directly result in a higher priority for expediting. However, for the other items in Table 47 a clear rise in priority is noted. This means that the suppliers from which these items are ordered will be contacted earlier. According to the supply chain manager, the expedite team is capable of expediting around 50 items each day. This means that for example the third item in Table 47 would be expedited around 6 days earlier due to shifting up 312 places on the priority list. The most impact will be made on the OTIF rate by the items which are not yet overdue, but are predicted to be delivered late, and are highly prioritized. These items can still be prevented from being delivered late, and thus could still be delivered on-time, which would increase the OTIF-rate.

It can be concluded from this future analysis, that the new prescriptive model allows Company X to act more proactively, due to having an estimate of when the ordered items will be delivered. It was seen that 207 items, 10.132% of the not yet overdue but highly critical items, would be added to the priority list for expediting by the new prescriptive model and can be prevented from being delivered late. This is manageable to be expedited within a couple of days by the expedite team, according to the supply chain manager of Company X. However, it is important to realize that the prescriptive model does not take the accuracy of the CNN into account. It was seen in Section 5.2.3 that the CNN had a mean absolute prediction error of 7.619 days. Hence, it could be that a number of these 207 items which would be overdue within two weeks and were predicted to be delivered late, are incorrectly predicted to be late and should actually not be included in the priority list for expediting. 132 Out of these 207 items are predicted to be delivered between 1 and 7 days late. Seeing the MAE of the CNN, it could be that those items would actually be delivered on-time and are being incorrectly added to the priority list.

Furthermore, it was seen that the highly critical items received a higher expedite score than originally in case the predicted number of days the item would be delivered late was higher than the number of days the item was already overdue. Hence, there is a high probability that these items will be positioned higher on the priority list for expediting, as was seen in Table 47. This means the suppliers will be contacted earlier about these PO items by Company X. This could prevent late deliveries from happening for the items which are not yet overdue, or from being delivered extremely late for the items which are already overdue. This in turn reduces risk for Company X, since those items are very critical to be able to perform the maintenance activities, and are of a high order value. However, a critical note needs to be placed about the reliability of the expedite score. Within the expedite score, the score for the factor days overdue was changed into the maximum of (Today - planned delivery date) and (predicted delivery date - planned delivery date). If the planned delivery date is not adjusted by the vendor, the value for (predicted delivery date - planned delivery date) equals (predicted delivery date - contractual delivery date) and is thus directly the outcome of the CNN. Then, if the PO item is not overdue yet, the result from the CNN, is directly taken as basis for the 'days overdue' score. The days overdue score is then raised with 1 point for each 2 days the PO item is predicted to be late, to a maximum of 10. Therefore, it is important to check how accurate the CNN is for cases with a target value in the range of (0,20), since an outcome in this range determines the height of the 'days overdue' score. The results for the CNN (trained on the total dataset) in this particular range, referred to as the cases for expediting, are given in Table 48.

Table 48: MAE performance of CNN for expediting

Case type	Target value range	Number of prefixes	MAE (days)	95% CI
Expedite	$(0, 20)$	677065	7.159	[7.140,7.179]

It can be observed that the MAE value for the prefixes in the test partition of the total dataset is 7.159 days. This means that the actual delivery date for the expedite cases is on average around 7 days earlier or later than the predicted delivery date as generated by the CNN. What could happen then for example is that a PO item is predicted to be delivered 13 days late. Therefore, the factor days overdue will get a score of 6, which then increases the expedite score with 0.20∗6=1.20. However, it seemed that the model was actually 7 days off and the item was actually delivered 20 days late. The days overdue score should then have been 10, increasing the expedite score with 2. The PO item then probably received a lower priority for expediting than it actually should have gotten. The prediction error can thus lead to biased values for the newly defined expedite score, and hence lead to a bias within the expedite priority list generated by the prescriptive model. Especially underpredicting the delivery date would be of negative impact for Company X, since it would mean that a PO item would get a lower priority for expediting than it should get and through which the item might not be prevented from arriving late. On the other hand, the factor 'days overdue' in which the predicted delivery date is included, determines the expedite score only by 20%. This limits the possible bias created by the prediction error within the overall expedite score and hence prescriptive model.

7 Conclusion and Discussion

This final chapter describes the conclusion of this master thesis research by answering the main research question. Furthermore, the limitations regarding the research are discussed. Next, recommendations are provided for Company X. Finally, suggestions are given for future research.

7.1 Conclusion

This research aims to increase transparency in the P2P process of Company X and improve decision-making with respect to expediting, in order to improve OTIF delivery performance for purchased items. In particular, this research addresses the following two main problems: Company X wants to (1) be able to act proactively and intervene before a PO item is actually delivered too late by a vendor, and (2) to increase the impact of expediting on the OTIF rate by improving the decision making on which items to prioritize for expediting. This leads to the following main research question:

How can the expected delivery date for purchased items be predicted and actions be recommended regarding expediting in order to increase the OTIF rate?

The main research question relates to the two main problems explained above. Firstly, it was investigated how Company X can be enabled to act proactively during the predictive part of this research, focused on finding the expected delivery date. Secondly, in order to improve the decision-making at Company X, a prescriptive part was added that aimed at finding which actions to recommend regarding expediting. The predictive part needed to be developed first, where subsequently the prescriptive part could be build further upon. The approach considered for the predictive part was based upon DL-based PBPM techniques and for the prescriptive part upon PrBPM techniques. In order to answer the main research question, eight sub questions needed to be answered.

SQ1. *Which data attributes are available, and how can additional features be engineered from these attributes, that might be influential on delivery performance and can be used as input for the predictive algorithm(s) to predict the expected delivery date?*

Before building the actual predictive models, an event log needed to be created including all relevant features to be used as input for the models to predict the expected delivery date. The final input data set consisted of the following selected features: the P2P process activities, time since case start, time since last event, time since midnight, weekday, vendor, plant, BU, item, material/part number, OTIF of last year, PO lines per vendor, number of changes done to the delivery date, and number of days between PO creation and the contractual delivery date. Besides, the model prediction target was defined as follows: the number of days between the contractual delivery date and goods receipt event (actual delivery date). The expected delivery date could then be obtained by adding this number of days to the contractual delivery date.

SQ2. *Which DL-based PBPM techniques are most appropriate to predict the expected delivery date for purchased items by Company X?*

The predictive models were selected from the set of DL-based PBPM techniques explored in the literature review (van Wijlen, 2022), based upon the examined P2P process characteristics and selected input features. First, the MLP approach of Venugopal et al. (2021) was selected due to it being a strong baseline frequently outperforming more sophisticated NNs. Furthermore, the LSTM approach of Tax et al. (2017) was included, since it is especially designed to deal with temporally dependent data and thus can exploit the temporal structure of activities in the event log. Additionally, the CNN from Pasquadibisceglie et al. (2020) was used in the research. This NN was selected since it has shown exceptional accuracy whenever applied to image data. Therefore, it seemed promising to apply to non-visual sequential data from traces in the event log. On top of that, two graph convolutional networks were selected: the GCN (Venugopal et al., 2021) and BIG-DGCNN (Chiorrini et al., 2021). These NNs can exploit the process structure by using process models as input. Hence, behavioral patterns, which are flattened in the event log, are better detected, which in turn can improve predictive performance. The best process model mined with the IMf filtering 40% of the noise, showed interesting enough behavioral patterns for the P2P process, like parallels and choices, from which the graph neural networks can benefit. Furthermore, the BIG-DGCNN uses special time features taking into account the process structure, which should enhance the models ability to detect temporal relations among activities in the event log. Finally, a Random Forest approach was added to these selected NN approaches. This approach investigated whether first classifying the extent to which a PO item would be delivered late/early with a RF followed by predicting the exact delivery date with the best performing NN would mitigate the negative effects on predictive performance due to extreme values for the target variable.

SQ3. *Which pre-processing steps need to be executed and which encoding techniques are most appropriate for the input data and chosen predictive algorithms, and how can these pre-processing and encoding steps be performed effectively?*

In order for the predictive algorithms to understand the input data and use it effectively, the input data set had to be further preprocessed. First, the categorical features were encoded by using either one-hot encoding or embedding. Furthermore, to prevent skewness from influencing model results, the values for the numerical input features were log-scaled. As the total dataset consists of approximately 1.2M cases, it was decided to create subsamples of different sizes from the total dataset. Hence, computational time was reduced for training and validating the predictive models for different configurations. Next, for all samples, the input data was transformed into a prefix format. Two sequence encoding methods were used. Prefix padding was used for the MLP, GCN, and LSTM, whereas aggregation encoding was used for the CNN and RFs. Subsequently, for evaluation purposes, the prefix data for all samples was split into a train, validation and test partition. Lastly, the prefix data was randomly undersampled and oversampled for certain experiments of the MLP, LSTM, and RFs. The last step during data preparation was to export the prepared sample datasets to CSV-files. These could then be easily imported like a data frame into Python within the model experiments.

SQ4. *How do the selected predictive algorithms need to be redesigned and built to be able to predict the expected delivery date for PO items from the event log and chosen input features?*

Although the five NNs were based on existing approaches within literature, several modifications needed to be performed to make the models work for the input dataset and prediction task. Commonly, the procedure had to be changed for feature and target vector creation due to having different features and another target. Moreover, often the creation of a validation partition had to be added to the splitting procedure. Furthermore, frequently changes had to be made to the model due to the original model having a classification instead of regression prediction type. The design of the RF approach was changed by using various types RF classifiers, different classes to predict, and hyper-parameter optimization for the model parameters.

SQ5. *How well do the built predictive algorithms predict the expected delivery date for PO items in terms of performance measures?*

The two best performing NNs in terms of MAE were found to be the CNN and BIG-DGCNN. When trained on the total dataset the CNN had a MAE of 7.619 days and BIG-DGCNN of 5.867 days on the test partition. Although, the BIG-DGCNN seemed more promising in terms of prediction accuracy, the CNN was chosen to be the best performing NN, due to being more suitable for implementation. It had a faster training and prediction generation procedure, as well as smaller memory consumption. Furthermore, the Balanced RF was found to perform the best of all tested RFs. However, the RF was not precise enough to implement as classifier for Company X. This was due to imbalanced data. Therefore, it was decided to just implement the CNN and train it on all the cases.

SQ6. *Which data attributes are available, and how can additional features be engineered from these attributes, that might be influential on whether a PO item from a supplier needs to be expedited or* *not?*

Subsequently, the generated predictions by the CNN could be used as input for the prescriptive model. The prescriptive task was defined as generating a priority for expediting (expedite score), for each of the PO items, based upon the predicted delivery date and other relevant features. These other relevant features were based on the existing expedite framework of Company X and were the following: material criticality, stock versus non-stock, days overdue score, OTIF of last month, and OTIF of last year. Values for these features were checked on data quality issues by means of a univariate analysis. No major issues were found. Hence, these features including the predictions could be used as input for the prescriptive model.

SQ7. *What PrBPM technique is able to generate recommendations for which PO items from suppliers to expedite and how are these recommendations achieved?*

The newly developed prescriptive model was based upon the existing expedite framework of Company X which consists of an expedite dashboard, showing the prescriptive input data as well as generated priorities for the PO items, and weighed scoring model to calculate a expediting priority score. In order to enable Company X to act proactively regarding purchased items and to improve their decision making with respect to the expediting process, several changes were made to existing expedite framework. This finally resulted in the newly developed prescriptive model. First of all, the range of items being expedited was changed. An extra filter was added to the dashboard such that not only the already overdue items could be selected, but also the items that will become overdue within the upcoming 14 days (when the planned delivery date is passed) and are predicted to be more than 1 day late. Hence, the expedite score would be calculated for a larger number of PO items, including items that are not yet overdue. Subsequently, for all these items a new expedite score was calculated where the factor ‘days overdue’ was changed such that it took the predicted delivery date into account as produced by the CNN. By changing this factor in the original framework, the expedite score took different values for the PO items which resulted in a new and different priority list for expediting.

SQ8. *How well is the PrBPM technique able to generate recommendations for which PO items from suppliers to expedite in terms of performance measures?*

The performance of the prescriptive model was evaluated by means of a historical and future analysis. The past analysis showed that for around 77% of the past expedited items it could have been predicted ahead of time that the items would arrive late. If the new prescriptive model was implemented at that time, these items could have been expedited before they would already be overdue. Furthermore, the future analysis showed that around 10% of the not yet delivered most critical purchased items would become overdue soon and were predicted to be delivered late. With implementing the new prescriptive model, these items are added to the expedite priority list, as well as assigned a higher priority for expediting in case the predicted delivery date is later than the planned delivery date. This could prevent late deliveries from happening or from being very extreme by contacting suppliers earlier. This reduces risk since those items are very critical to be able to perform the maintenance activities and are of high order value. A critical note to the prescriptive model is that it does not take into account the accuracy of the CNN. Therefore, items could be incorrectly included into the expediting priority list as they were incorrectly predicted to be delivered too late. Moreover, it affects the reliability of the new expedite score and can lead to a bias within the generate expediting priority list.

To summarize, the results show that by implementing the CNN as well as the new prescriptive model using the generated predictions by the CNN, Company X is enabled to act proactively and intervene before items are delivered late. The CNN creates insight into which items are expected to be delivered late. Hence, by using the new prescriptive model these items can be included within the priority list for expediting and a higher priority can be assigned to items which are predicted to arrive later than the planned delivery date. Therefore, Company X can call suppliers before the items are actually overdue. Moreover, by giving more priority to items for expediting which are not overdue yet but are predicted to arrive very late, impact can be made on the OTIF rate since those items can be prevented from actually arriving late or from arriving extremely late.

7.2 Limitations

Several limitations exist with respect to this master thesis research.

Hyper-parameter optimization. The first limitation is related to the fact that no hyper-parameter optimization was performed to retrieve optimal values for the hyper-parameters of the NNs. The hyper-parameter values were solely based upon the values taken within in the original approaches for the five NNs. However, these original approaches for the five NNs all had another prediction task than we had, as well as used other and/or less input features. Therefore, it cannot be guaranteed that those values were optimal for our prediction task and input features. For example, one of the elements that determines the values for the number of neurons in the hidden layers is the complexity of the input dataset (Krishnan, 2021). As our prefix input dataset contains more features than the input datasets used within the researches of the original NN approaches, it could be that the number of neurons in the original approach is not sufficient enough to understand the input and hidden relations in the data, and possibly predictive performance could have been improved by using a higher number of neurons.

Min-max normalization. Furthermore, it was seen that min-max normalization was applied to the input features after log-scaling only for the CNN, as this was done in ther original approach of Pasquadibisceglie et al. (2020). This normalization shifts the feature values closer to the mean and gives smaller standard deviations. Moreover, it suppresses the weight of outliers in the data (Lindgren, 2019). Since this was only applied for the CNN it is hard to conclude whether the CNN is performing better than most of the other models because of its architecture or due to this extra normalization.

Subsamples. Another limitation is related to the fact that the two best NNs were selected based on the results for subsample $k = 6$, and not the results for the total dataset, as not all models were tested using the total dataset. Although the distribution of the target values within subsample $k = 6$ seemed fairly similar to the distribution of the total dataset by looking at the distribution plots and CDF, no statistical tests were used to prove that subsample $k = 6$ was a good representative of the total dataset. Moreover, not only the distribution of the target variable could influence predictive performance, but also the distribution of the values for the input features. It was not investigated whether the sampling procedure actually keeps the distribution of the various input features similar to that of the total dataset. Therefore, it is questionable whether it can be assumed that the performance of the NNs on subsample $k = 6$ is similar to the performance on the total dataset.

Minimum prefix length. On top of that, with the final comparison of the CNN and BIG-DGCNN, it is a limitation that the CNN uses a smaller minimum prefix length than the BIG-DGCNN. The shortest prefix that is included in the CNN approach is a prefix of 1 P2P event, whereas for the BIG-DGCNN this is a prefix including 1 artificial start event and 2 P2P events for the BIG-DGCNN. This influences the overall MAE calculated over the prefixes in the test set. For example, if the prefixes with length 1 were excluded from the test set, the overall MAE would have been 7.345 instead of 7.619 days. However, as shown, this difference in MAE is not large, and the predictive accuracy is still clearly better for the BIG-DGCNN, with 5.867 days.

Training data. Another limitation is related to the data selected for training the predictive models. Since data was available within Celonis from the year 2019 and it was suggested by the supply chain manager of Company X to include all past data for training, it was assumed that this was a good practice. However, it would have been better to check how the number of days between contractual delivery date and actual delivery date changed over the years 2019-2022 to check whether the P2P process changed to a great extent over the years. For example, it could have been the case that there were more late deliveries, or more extreme late deliveries, due to the COVID-19 crisis within 2020-2021. This might not reflect the usual process behavior and distort the predictive performance on the more recent data. On the other hand, the distribution of target variable, i.e. the number of days between contractual delivery date and actual delivery date, was compared for the train, validation, and test partition of subsample $k = 6$. The train, validation, and test set partition represent respectively the past, more recent, and most recent data as the split was temporally based. Their target value distributions looked highly similar. However, it cannot be guaranteed that this is the case for the total dataset.

Future analysis. Then, with respect to the prescriptive process monitoring phase there is a limitation regarding the future analysis used to evaluate the prescriptive model. The future analysis was only performed at one moment in time. However, the items that are included are the items that are due within 14 days from the moment of analysis, which means that this selection of items can change every day. Hence, it would have been more reliable to perform the analysis at multiple fixed moments in time within a longer time period.

Accuracy of prediction model. The last limitation considers the fact that when the CNN is deployed, and predictions are generated for the not yet delivered PO items, no indication is provided for each prediction of its accuracy. Hence, the prescriptive model cannot directly account for a possible prediction error while generating the expediting priority list.

7.3 Recommendations

There are several recommendations for Company X to improve the OTIF rate for items ordered from external suppliers.
First of all, it is recommended to start a validation period in which the deployed CNN and prescriptive model are used to steer the expedite process. This way, their impact can be validated with respect to the OTIF rate. It is recommended to start with validating the CNN and prescriptive model only for PO items from one business unit. When validating the implemented CNN it is important to keep monitoring the predictions for the open PO items up until the moment they are delivered in full. Hence, the prediction error can be tracked for the different items over the course of the P2P process. This monitoring can for example be done by taking a snapshot of the predictions for all items every week and store them in an Excel sheet, or to monitor it every day and store the results automatically within an Excel sheet by creating an action flow in Celonis. Monitoring the prediction error could for example show whether more accurate predictions are generated when more events have happened for the PO item. Furthermore, it helps detecting concept drift. When the business environment changes, concept drift can arise in prediction models. This is when the relation between the feature and target variables changes due to external factors, which causes a decrease of prediction accuracy over time. According to Park and Song (2020) predictive models should then be adapted in an online manner and need to be updated if prediction accuracy becomes too low. Therefore, a threshold needs to be determined for the prediction error to get an indication of when the CNN needs to be retrained. Besides, it is important to monitor the effect of the prediction error on the expedite score, to get an understanding of whether items are wrongly prioritized. This could indicate whether the new expedite score needs to be changed, for example by assigning more or less weight to the days overdue score which includes the predicted delivery date. Moreover, it is valuable to keep track of how often PO items are expedited which are not yet overdue, but are predicted to become overdue. This gives an indication of the extent to which the prescriptive model allows Company X to act more proactively regarding the delivery of PO items. Furthermore, the OTIF needs to be monitored during the validation period to validate the actual impact of the CNN together with the prescriptive model.
When the models are further fine-tuned based on the results of this validation period, Company X can start extending the deployment of the models by validating them for more business units. When sufficient confidence is gained about that the implementation of the CNN and prescriptive model actually improves the OTIF rate, the models can be fully enrolled.

7.4 Future Research

This research is completed with giving directions for future research.

Encoding high cardinality features. First of all, it is suggested to investigate other methods than one-hot encoding to encode categorical variables with high cardinality. Some attributes, like vendor and material/part number, had such a high cardinality that the number of dummy features by which the attribute was represented was insufficient to capture the relations for those features, as 99% of the values was placed in the category 'others'. Hence, these features were almost not taken into account when using one-hot encoding. Therefore, embedding was tested as well, however, predictive performance remained the same or got worse. Since, it was only tested for the MLP and LSTM, it is suggested to further explore the embedding technique. Moreover, regularized target encoding has shown promising performance for encoding high cardinality features to be used in ML techniques within Pargent et al. (2022), and its application could be explored for DL-based PBPM.

Over/Undersampling. As the predictive performance of the NNs was suffering from the presence of rare and extreme continuous target values and research is lacking on the topic of imbalanced data for regression, it is suggested to investigate whether more sophisticated over- and undersampling techniques can be applied to prefix input data to overcome this problem. For example, a generative adversarial network (GAN) is a recent and upcoming technique for data augmentation to solve the problem of having imbalanced image classification as it is capable of generating realistic samples from the input on which the model is trained Brownlee (2019). Hence, it could be an interesting method to explore for oversampling prefixes.

RF approach. Another direction for future research is to further look into the RF approach, where the extent of late/early delivery is first classified by a RF and subsequently an exact delivery date is predicted by a NN trained on the specific type of case as which the case was classified. This could be a good option to overcome the problem of diminished prediction accuracy due to imbalanced data. Therefore, it is suggested to further explore how the RFs can be improved, or to investigate more sophisticated models from the field of deep learning.

BIG-DGCNN. Moreover, since the BIG-DGCNN from Chiorrini et al. (2022) showed highly promising results in terms of prediction accuracy, but lacked computational efficiency, it is suggested to further explore how this approach can be changed such that it becomes less computationally intensive and can be used more easily on large datasets.

Accuracy of prediction model. Lastly, it is proposed to examine how the deployment of a PBPM method can be enhanced by providing an indication the reliability of each prediction generated by the model. For example, it can be investigated whether it is possible to efficiently find the prefixes in a historic dataset that are most alike the running case and give a confidence interval for the prediction error based on the performance of the model on these historic prefixes.

References

Aalst, W. v. d., Adriansyah, A., Medeiros, A. K. A. d., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., Brand, P. v. d., Brandtjen, R., Buijs, J., et al. (2011). Process mining manifesto. In *International conference on business process management*, pages 169–194. Springer.

Anaplan (2022). Understand advanced metrics. https://help.anaplan.com/685ff9b2-6370-46ba-af10-679405937113-Understand-advanced-metrics.

Arat, M. M. (2019). How to use embedding layer and other feature columns together in a network using keras? https://mmuratarat.github.io/2019-06-12/embeddings-with-numeric-variables-Keras.

Becker, J., Brunk, J., Ding, W., and Niemann, M. (2020). Conceptualization of an integrated procedure model for business process monitoring and prediction. In *2020 IEEE 22nd Conference on Business Informatics (CBI)*, volume 1, pages 49–57. IEEE.

Bergstra, J., Yamins, D., and Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR.

Brownlee, J. (2019). A gentle introduction to generative adversarial networks (gans). https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/.

Brownlee, J. (2020). What is argmax in machine learning? https://machinelearningmastery.com/argmax-in-machine-learning/.

Celonis (2019). Celonis process mining demo: Machine learning. Available online at: https://www.youtube.com/watch?v=FJoE2Dbi7Dw&t=145s, last accessed on 2022-04-28.

Celonis Inc. (2022). Machine learning workbench. https://docs.celonis.com/en/machine-learning-workbench.html.

Chiorrini, A., Diamantini, C., Genga, L., and Domenico, P. (2021). Multi-perspective enriched instance graphs for next activity prediction through graph neural network.

Chiorrini, A., Diamantini, C., Mircoli, A., and Potena, D. (2022). Exploiting instance graphs and graph neural networks for next activity prediction. In *International Conference on Process Mining*, pages 115–126. Springer.

Claes, Jan (2022). Add artificial events plugin. https://www.janclaes.info/post.php?post=addartificialevents.

Courthoud, M. (2022). How to compare two or more distributions. https://towardsdatascience.com/how-to-compare-two-or-more-distributions-9b06ee4d30bf.

Curran-Everett, D. (2017). Explorations in statistics: the assumption of normality. *Advances in physiology education*, 41(3):449–453.

Data Science Process Alliance (2022). What is crisp dm? Available online at: https://www.datascience-pm.com/crisp-dm-2/, last accessed on 2022-05-17.

Databricks (2022). Best practices: Hyperparameter tuning with hyperopt. https://docs.databricks.com/machine-learning/automl-hyperparam-tuning/hyperopt-best-practices.html.

de Leoni, M., Dees, M., and Reulink, L. (2020). Design and evaluation of a process-aware recommender system based on prescriptive analytics. In *2020 2nd International Conference on Process Mining (ICPM)*, pages 9–16. IEEE.

Diamantini, C., Genga, L., Potena, D., and van der Aalst, W. (2016). Building instance graphs for highly variable processes. *Expert Systems with Applications*, 59:101–118.

Ektamaini (2020). Z score for outlier detection – python. `https://www.geeksforgeeks.org/z-score-for-outlier-detection-python/`.

Fahrenkrog-Petersen, S. A., Tax, N., Teinemaa, I., Dumas, M., Leoni, M. d., Maggi, F. M., and Weidlich, M. (2021). Fire now, fire later: alarm-based systems for prescriptive process monitoring. *Knowledge and Information Systems*, pages 1–29.

Fani Sani, M., Vazifehdoostirani, M., Park, G., Pegoraro, M., Zelst, S. J. v., and van der Aalst, W. M. (2021). Event log sampling for predictive monitoring. In *International Conference on Process Mining*, pages 154–166. Springer, Cham.

Fath, A. H., Madanifar, F., and Abbasi, M. (2020). Implementation of multilayer perceptron (mlp) and radial basis function (rbf) neural networks to predict solution gas-oil ratio of crude oil systems. *Petroleum*, 6(1):80–91.

Fraunhofer FIT (2021a). Handling event data. `https://pm4py.fit.fraunhofer.de/documentation#importing`.

Fraunhofer FIT (2021b). Process discovery. `https://pm4py.fit.fraunhofer.de/documentation#discovery`.

Frost, J. (2019). Using confidence intervals to compare means. `https://statisticsbyjim.com/hypothesis-testing/confidence-intervals-compare-means/`.

Gawali, S. (2021). Shape of data: Skewness and kurtosis. `https://www.analyticsvidhya.com/blog/2021/05/shape-of-data-skewness-and-kurtosis/`.

Glen, S. (2022a). Anova test: Definition, types, examples, spss. `https://www.statisticshowto.com/probability-and-statistics/hypothesis-testing/anova/`.

Glen, S. (2022b). Rmse: Root mean square error. `https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/`.

Harane, N. and Rathi, S. (2020). Comprehensive survey on deep learning approaches in predictive business process monitoring. *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*, pages 115–128.

Hinkka, M., Lehto, T., and Heljanko, K. (2018). Exploiting event log event attributes in rnn based prediction. In *Data-Driven Process Discovery and Analysis*, pages 67–85. Springer.

Huh, K. (2021). Surviving in a random forest with imbalanced datasets. `https://medium.com/sfu-cspmp/surviving-in-a-random-forest-with-imbalanced-datasets-b98b963d52eb`.

Intel (2022). A new mobile computing experience for business. `https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/8th-gen-core-vpro-processor-brief.pdf`.

Jalayer, A., Kahani, M., Beheshti, A., Pourmasoumi, A., and Motahari-Nezhad, H. R. (2020). Attention mechanism in predictive business process monitoring. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 181–186. IEEE.

Kanal, L. N. (2003). Perceptron. In *Encyclopedia of Computer Science*, pages 1383–1385.

Karra, S. (2022). Normalization in machine learning. `https://sailajakarra.medium.com/normalization-in-machine-learning-166a364d3edc`.

Khan, A., Le, H., Do, K., Tran, T., Ghose, A., Dam, H., and Sindhgatta, R. (2021). Deepprocess: Supporting business process execution using a mann-based recommender system. In *International Conference on Service-Oriented Computing*, pages 19–33. Springer.

Khan, N., Ali, Z., Ali, A., McClean, S., Charles, D., Taylor, P., and Nauck, D. (2019). A generic model for end state prediction of business processes towards target compliance. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 325–335. Springer.

Kirchmer, M. et al. (2017). *High performance through business process management.* Springer.

Klinkmüller, C., van Beest, N. R., and Weber, I. (2018). Towards reliable predictive process monitoring. In *International Conference on Advanced Information Systems Engineering*, pages 163–181. Springer.

Komorowski, M., Marshall, D. C., Salciccioli, J. D., and Crutain, Y. (2016). Exploratory data analysis. *Secondary analysis of electronic health records*, pages 185–203.

Krishnan, S. (2021). How do determine the number of layers and neurons in the hidden layer? https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3.

Kumar, A. (2022). One-hot encoding concepts & python examples. https://vitalflux.com/one-hot-encoding-concepts-python-code-examples/.

Langsrud, Ø. (2003). Anova for unbalanced data: Use type ii instead of type iii sums of squares. *Statistics and Computing*, 13(2):163–167.

LeanX (2022a). Sap table eban. Available online at: https://leanx.eu/en/sap/table/eban.html, last accessed on 2022-05-16.

LeanX (2022b). Sap table ekko. Available online at: https://leanx.eu/en/sap/table/ekko.html, last accessed on 2022-05-16.

Leemans, S. J., Fahland, D., and Van Der Aalst, W. M. (2013). Discovering block-structured process models from event logs containing infrequent behaviour. In *International conference on business process management*, pages 66–78. Springer.

Lindgren, I. (2019). Transformations, scaling and normalization. https://medium.com/@isalindgren313/transformations-scaling-and-normalization-420b2be12300.

Maggi, F. M., Francescomarino, C. D., Dumas, M., and Ghidini, C. (2014). Predictive monitoring of business processes. In *International conference on advanced information systems engineering*, pages 457–472. Springer.

Matplotlib development team (2022). Plot types. https://matplotlib.org/stable/plot_types/index.html.

Mbaabu, O. (2020). Introduction to random forest in machine learning. https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/#:~:text=Advantages%20of%20random%20forest,over%20the%20decision%20tree%20algorithm.

Metzger, A., Franke, J., and Jansen, T. (2019). Data-driven deep learning for proactive terminal process management. In *BPM (Industry Forum)*, pages 190–201.

Mohammed, R., Rawashdeh, J., and Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (ICICS)*, pages 243–248. IEEE.

Mohtadi, B. F. (2017). In depth: Parameter tuning for random forest. https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d.

Muthukrishnan (2021). Understanding correlations and correlation matrix. https://muthu.co/understanding-correlations-and-correlation-matrix/.

Naderifar, V., Sahran, S., and Shukur, Z. (2019). A review on conformance checking technique for the evaluation of process mining algorithms. *TEM Journal*, 8(4):1232.

Neptune Labs (2022). How to deal with imbalanced classification and regression data. https://neptune.ai/blog/how-to-deal-with-imbalanced-classification-and-regression-data.

NetworkX Developers (2022). Tutorial. https://networkx.org/documentation/stable/tutorial.html.

NumFOCUS (2022a). Csv & text files. https://pandas.pydata.org/docs/user_guide/io.html#io-read-csv-table.

NumFOCUS (2022b). pandas.dataframe.groupby. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html.

NumFOCUS (2022c). pandas.get_dummies. https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html.

NumFOCUS (2022d). Pandas.qcut. https://pandas.pydata.org/docs/reference/api/pandas.qcut.html.

Pargent, F., Pfisterer, F., Thomas, J., and Bischl, B. (2022). Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *Computational Statistics*, pages 1–22.

Park, G. and Song, M. (2019). Prediction-based resource allocation using lstm and minimum cost and maximum flow algorithm. In *2019 international conference on process mining (ICPM)*, pages 121–128. IEEE.

Park, G. and Song, M. (2020). Predicting performances in business processes using deep neural networks. *Decision Support Systems*, 129:113191.

Pasquadibisceglie, V., Appice, A., Castellano, G., and Malerba, D. (2019). Using convolutional neural networks for predictive process analytics. In *2019 international conference on process mining (ICPM)*, pages 129–136. IEEE.

Pasquadibisceglie, V., Appice, A., Castellano, G., and Malerba, D. (2020). Predictive process mining meets computer vision. In *International Conference on Business Process Management*, pages 176–192. Springer.

Peltarion (2022). Macro-precision. https://peltarion.com/knowledge-center/evaluation-view/classification-loss-metrics/macro-precision.

Philipp, P., Georgi, R. X. M., Beyerer, J., and Robert, S. (2019). Analysis of control flow graphs using graph convolutional neural networks. In *2019 6th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, pages 73–77. IEEE.

Python Software Foundation (2022). argparse - parser for command-line options, arguments and sub-commands. https://docs.python.org/3/library/argparse.html.

PyTorch Contributors (2022a). Pytorch documentation. https://pytorch.org/docs/stable/index.html.

PyTorch Contributors (2022b). Torch.optim.optimizer.zero_grad. https://pytorch.org/docs/stable/generated/torch.optim.Optimizer.zero_grad.html.

Rama-Maneiro, E., Vidal, J., and Lama, M. (2021a). Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Transactions on Services Computing*.

Rama-Maneiro, E., Vidal, J. C., and Lama, M. (2021b). Embedding graph convolutional networks in recurrent neural networks for predictive monitoring. *arXiv preprint arXiv:2112.09641*.

Rančić, S., Radovanović, S., and Delibašić, B. (2021). Investigating oversampling techniques for fair machine learning models. In *International Conference on Decision Support System Technology*, pages 110–123. Springer.

Rzad, A., Wojnecka, J., Rutkowski, M., and Guliński, M. (2019). Investigating purchase-to-pay process using process mining in a multinational corporation.

SchedMD (2021). Slum workload manager: Overview. `https://slurm.schedmd.com/overview.html`.

scikit-learn developers (2022a). Randomforestclassifier. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`.

scikit-learn developers (2022b). Selectfrommodel. `https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html`.

scikit-learn developers (2022c). Sequentialfeatureselection. `https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html`.

scikit-learn developers (2022d). Stratifiedkfold. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html`.

scikit-learn developers (2022e). User guide. `https://scikit-learn.org/stable/user_guide.html`.

Shixin, L. (2020). Exploratory data analysis, feature selection for better ml models. `https://cloud.google.com/blog/products/ai-machine-learning/building-ml-models-with-eda-feature-selection`.

Shmueli, B. (2019). Multi-class metrics made simple, part ii: the f1-score. `https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1`.

Shoush, M. and Dumas, M. (2021). Prescriptive process monitoring under resource constraints: A causal inference approach. *arXiv preprint arXiv:2109.02894*.

Sola, J. and Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on nuclear science*, 44(3):1464–1468.

Stack Exchange Inc (2020). Macro- or micro-average for imbalanced class problems. `https://datascience.stackexchange.com/questions/36862/macro-or-micro-average-for-imbalanced-class-problems`.

Sukumar, R. (2020). Introduction to automatic hyperparameter optimization with hyperopt. `https://medium.com/analytics-vidhya/introduction-to-automatic-hyperparameter-optimization-with-hyperopt-e0b9c84d1059`.

SURF (2022). Snellius: de nationale supercomputer. `https://www.surf.nl/snellius-de-nationale-supercomputer`.

Tarang, S. (2017). About train, validation and test sets in machine learning. `https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7`.

Tax, N., Teinemaa, I., and van Zelst, S. J. (2018). An interdisciplinary comparison of sequence modeling methods for next-element prediction. *arXiv preprint arXiv:1811.00062*.

Tax, N., Verenich, I., Rosa, M. L., and Dumas, M. (2017). Predictive business process monitoring with lstm neural networks. In *International Conference on Advanced Information Systems Engineering*, pages 477–492. Springer.

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

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

Tensorflow (2021). Essential documentation. `https://www.tensorflow.org/guide`.

Tensorflow (2022a). tf.keras.callbacks.earlystopping. `https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping`.

Tensorflow (2022b). tf.keras.callbacks.modelcheckpoint. `https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint`.

Tensorflow (2022c). tf.keras.model. `https://www.tensorflow.org/api_docs/python/tf/keras/Model`.

The imbalanced-learn developers (2022a). Balancedrandomforestclassifier. `https://imbalanced-learn.org/stable/references/generated/imblearn.ensemble.BalancedRandomForestClassifier.html`.

The imbalanced-learn developers (2022b). Randomoversampler. `https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html`.

The imbalanced-learn developers (2022c). Randomundersampler. `https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html`.

Torgo, L., Ribeiro, R. P., Pfahringer, B., and Branco, P. (2013). Smote for regression. In *Portuguese conference on artificial intelligence*, pages 378–389. Springer.

Vadapalli, P. (2021). Random forest classifier: Overview, how does it work, pros cons. `https://www.upgrad.com/blog/random-forest-classifier/`.

Van der Aalst, W. M. (2016). *Process mining: data science in action*. Springer.

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 *International conference on application and theory of petri nets*, pages 444–454. Springer.

van Wijlen, M. (2022). A systematic literature review on predictive and prescriptive process monitoring: Exploring the state-of-the-art methods.

Venugopal, I., Töllich, J., Fairbank, M., and Scherp, A. (2021). A comparison of deep-learning methods for analysing and predicting business processes. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

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 (TIST)*, 10(4):1–34.

Visual Design (2021). Feature selection and eda in machine learning. `https://www.visual-design.net/post/feature-selection-and-eda-in-machine-learning`.

Wang, J., Yu, D., Liu, C., and Sun, X. (2019). Outcome-oriented predictive process monitoring with attention-based bidirectional lstm neural networks. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 360–367. IEEE.

Weinzierl, S. (2021). Exploring gated graph sequence neural networks for predicting next process activities. In *International Conference on Business Process Management*, pages 30–42. Springer.

Weinzierl, S., Dunzer, S., Zilker, S., and Matzner, M. (2020a). Prescriptive business process monitoring for recommending next best actions. In *International conference on business process management*, pages 193–209. Springer.

Weinzierl, S., Zilker, S., Brunk, J., Revoredo, K., Nguyen, A., Matzner, M., Becker, J., and Eskofier, B. (2020b). An empirical comparison of deep-neural-network architectures for next activity prediction using context-enriched process event logs. *arXiv preprint arXiv:2005.01194*.

Welcker, L., Koch, S., and Dellmann, F. (2012). Improving classifier performance by knowledge-driven data preparation. In *Industrial Conference on Data Mining*, pages 151–165. Springer.

Zach (2021). What does a high f value mean in anova? `https://www.statology.org/what-does-a-high-f-value-mean/`.

Appendix A

This appendix is related to Chapters 3 and 4. It shows more detailed results for the business and data understanding phase as well as for the process discovery and data analysis.

Process Models

The process models generated by the HM and IMf filtering 40% of the noise are depicted in Figures 35 and 36 respectively. Due to their large size the figures are located on the next page.

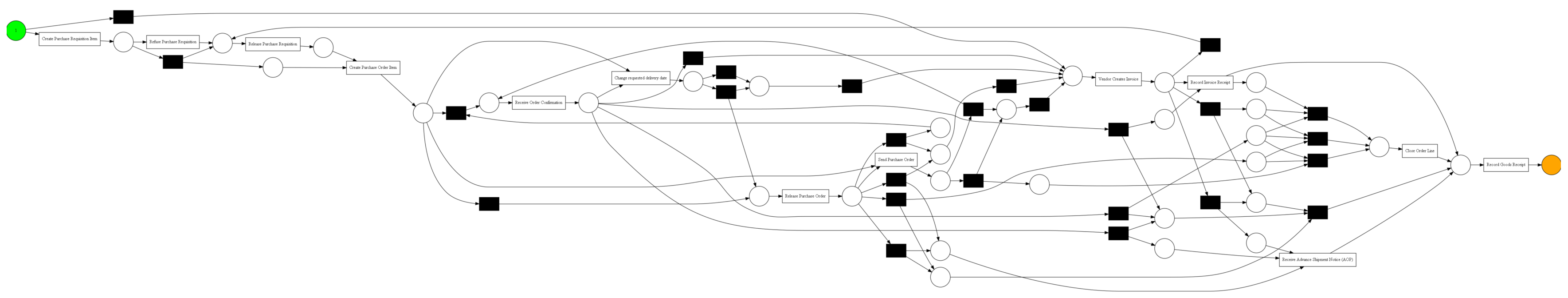

Create Purchase Requisition Item
Refine Purchase Requisition
Release Purchase Requisition
Create Purchase Order Item
Receive Order Confirmation
Change requested delivery date
Release Purchase Order
Send Purchase Order
Vendor Creates Invoice
Record Invoice Receipt
Close Order Line
Record Goods Receipt
Receive Advance Shipment Notice (ASN)

Figure 35: Process Model discovered by the Heuristics Miner

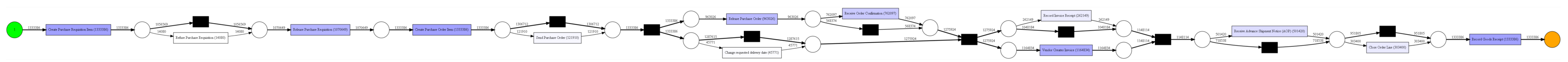


Figure 36: Process Model discovered by the Inductive Miner infrequent with 60% noise included

Distribution Plots of the Categorical Attributes

The probability distributions for the different categorical attributes are shown in Figures 37 - 44.

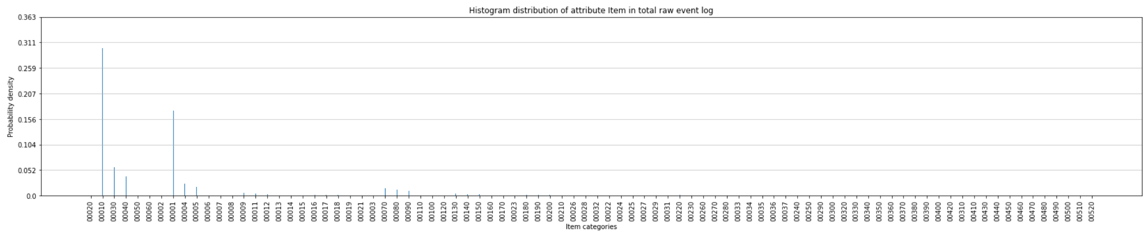


Figure 37: Distribution plot of attribute Item

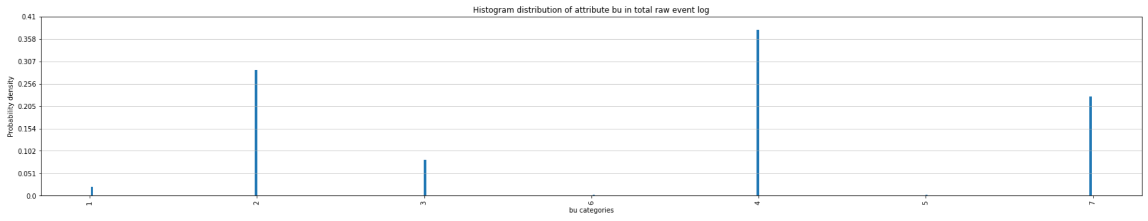


Figure 38: Distribution plot of attribute BU

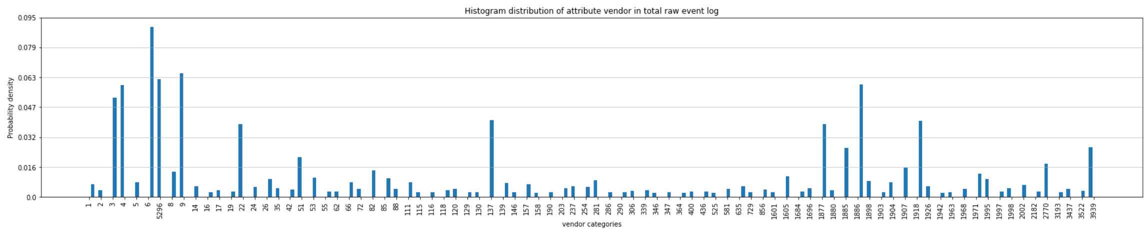


Figure 39: Distribution plot of attribute Vendor

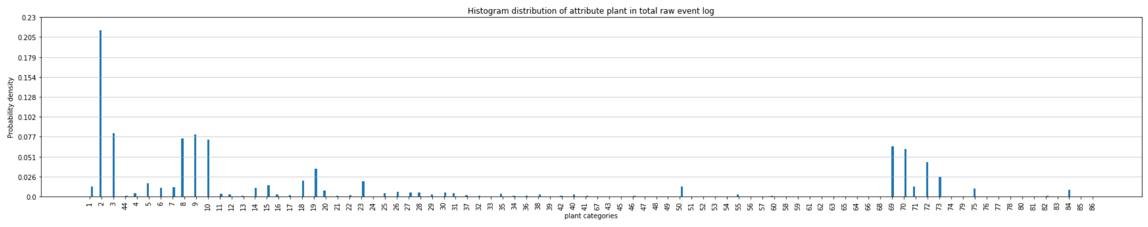


Figure 40: Distribution plot of attribute Plant

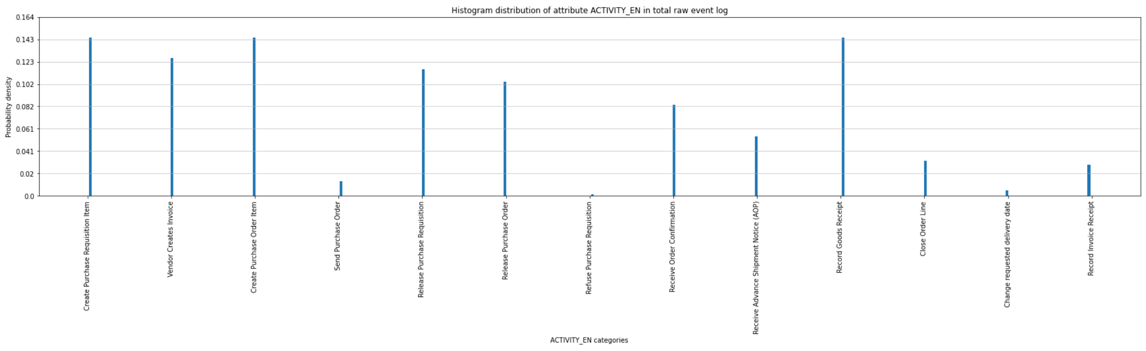


Figure 41: Distribution plot of attribute Activity

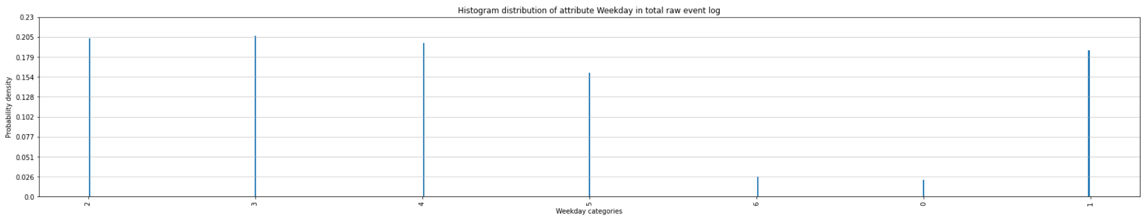


Figure 42: Distribution plot of attribute Weekday

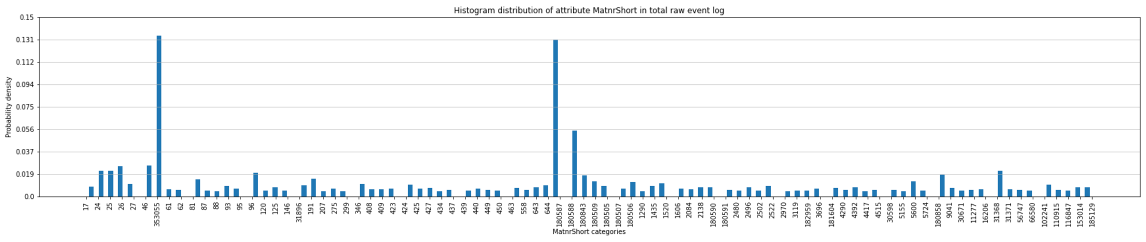


Figure 43: Distribution plot of attribute Material/Part number

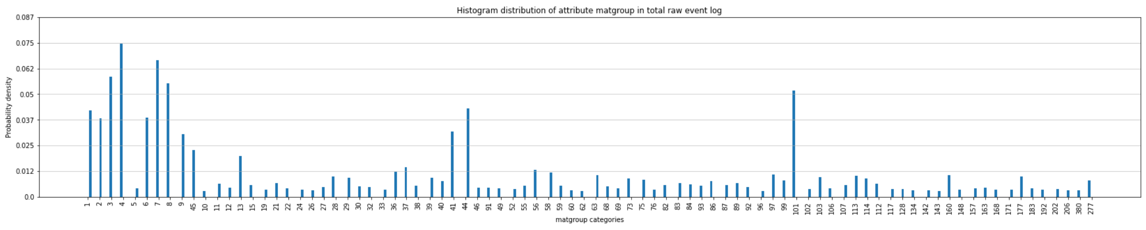


Figure 44: Distribution plot of attribute Material Group

Numerical Attributes

This section shows detailed figures for the relations among the numerical features and the relation between each of the features with the target variable.

Correlation Matrix

Figure 45 shows the correlation values for each pair of numerical features and for the numerical features and target variable.

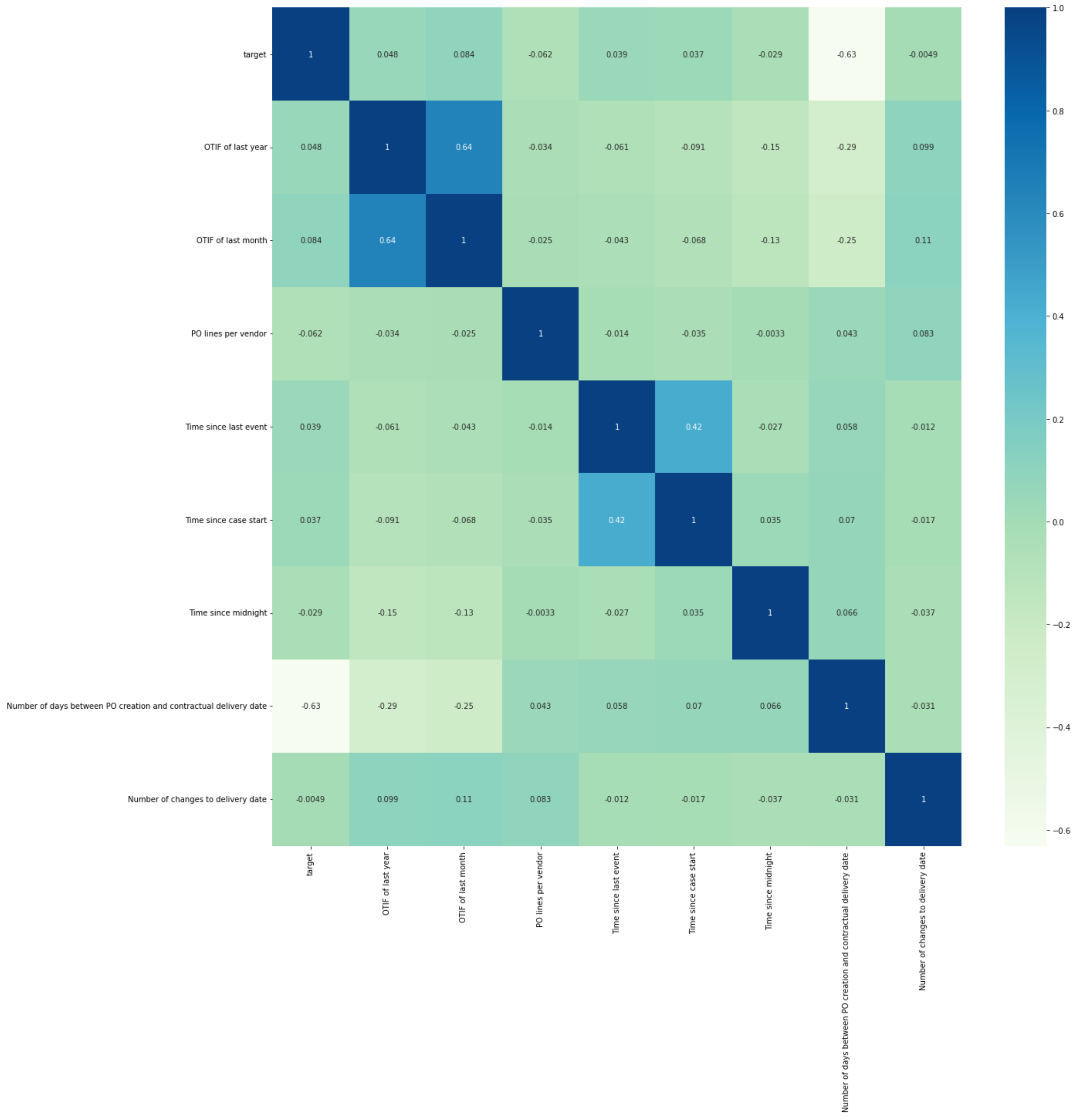


Figure 45: Correlation matrix numerical features

Scatterplots

Figures 46 - 61 depict the relation between the numerical attributes and the target variable. Two type of plots are provided: (1) target value has the range as in the raw dataset, and (2) target value has the range without outliers.

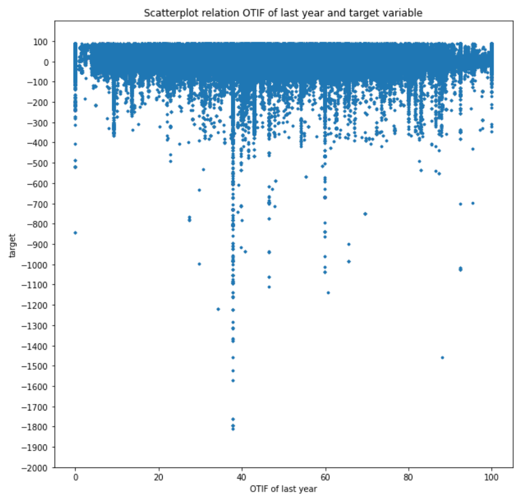


Figure 46: Scatterplot relation OTIF of last year and target variable

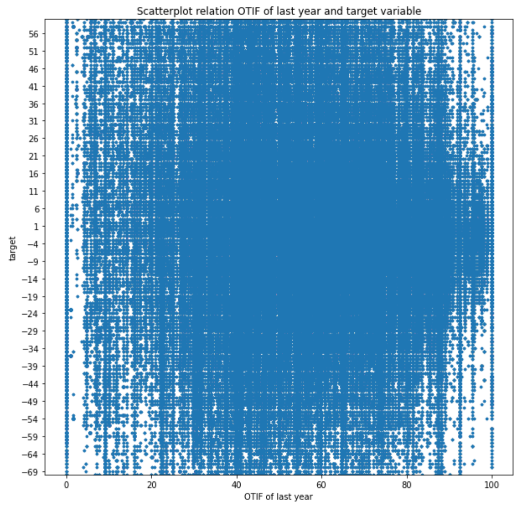


Figure 47: Scatterplot relation OTIF of last year and target variable - target range without outliers

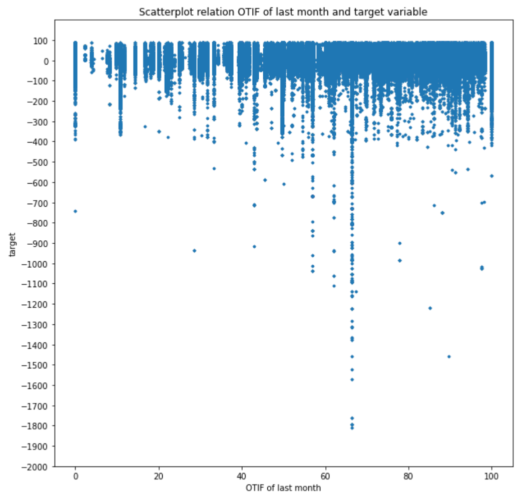


Figure 48: Scatterplot relation OTIF of last month and target variable

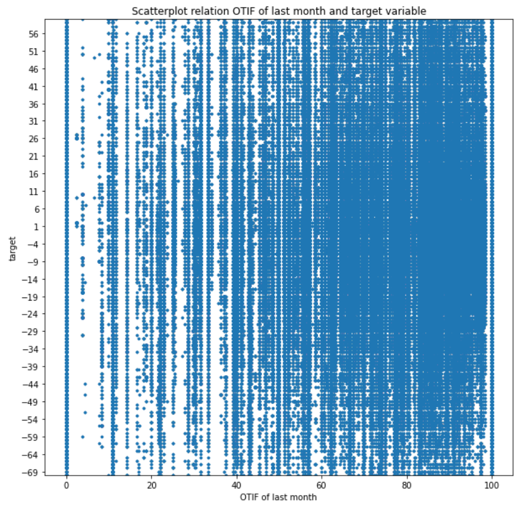


Figure 49: Scatterplot relation OTIF of last month and target variable - target range without outliers

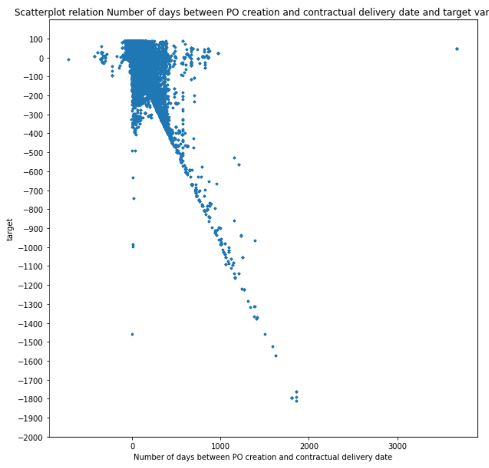


Figure 50: Scatterplot relation number of days between PO creation and contractual delivery date and target variable

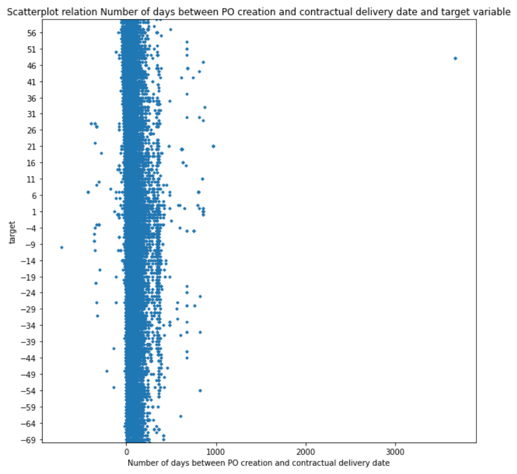


Figure 51: Scatterplot relation number of days between PO creation and contractual delivery date and target variable - target range without outliers

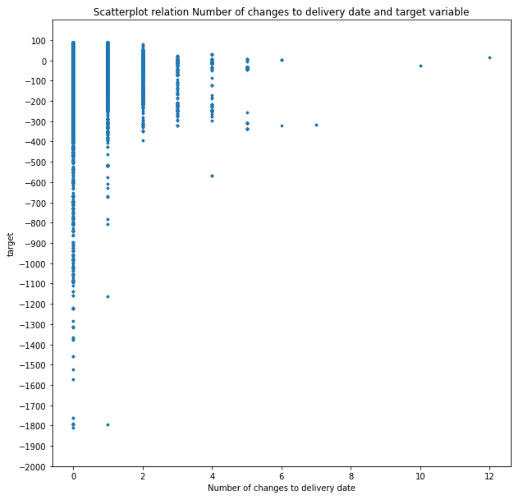


Figure 52: Scatterplot relation number of changes to delivery date and target variable

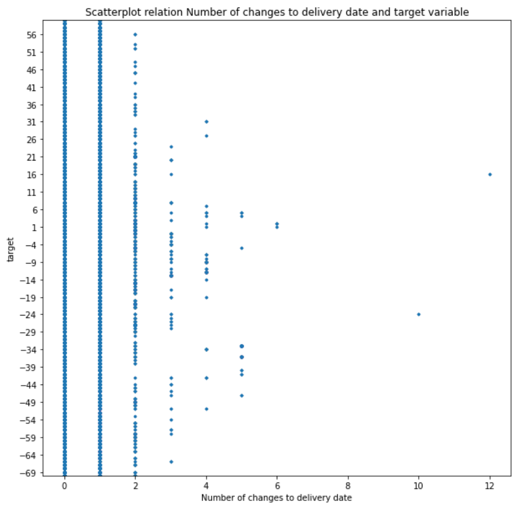


Figure 53: Scatterplot relation number of changes to delivery date and target variable- target range without outliers

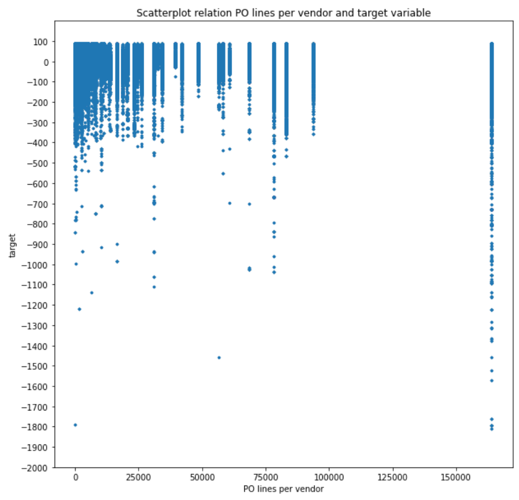


Figure 54: Scatterplot relation PO lines per vendor and target variable

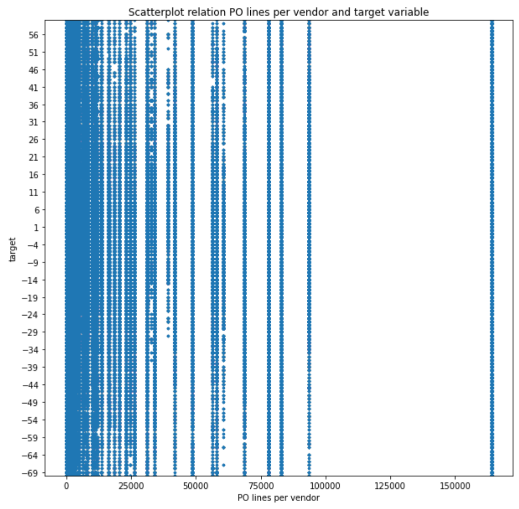


Figure 55: Scatterplot relation PO lines per vendor and target variable - target range without outliers

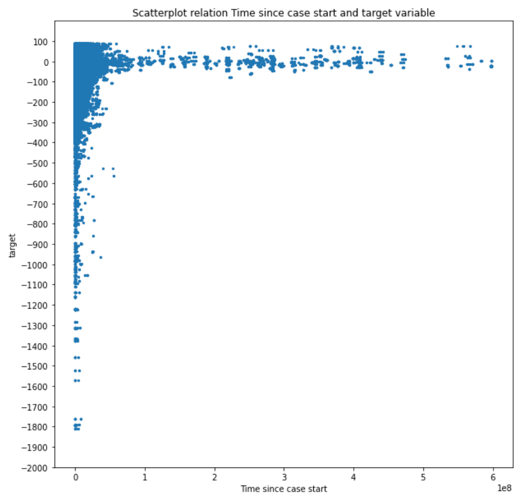


Figure 56: Scatterplot relation time since case start and target variable

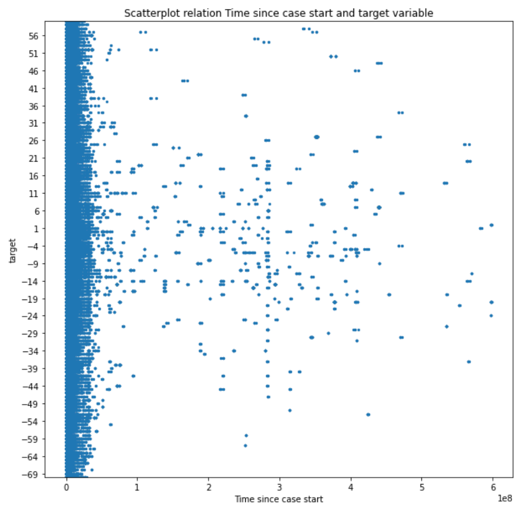


Figure 57: Scatterplot relation time since case start and target variable - target range without outliers

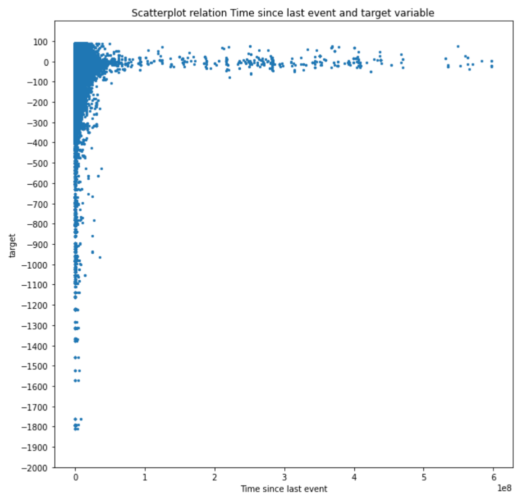


Figure 58: Scatterplot relation time since last event and target variable

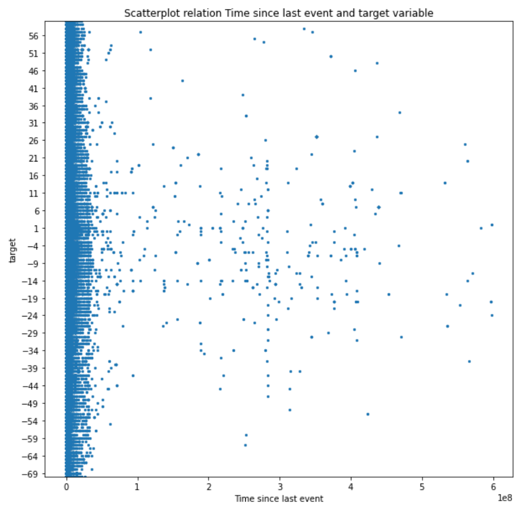


Figure 59: Scatterplot relation time since last event and target variable - target range without outliers

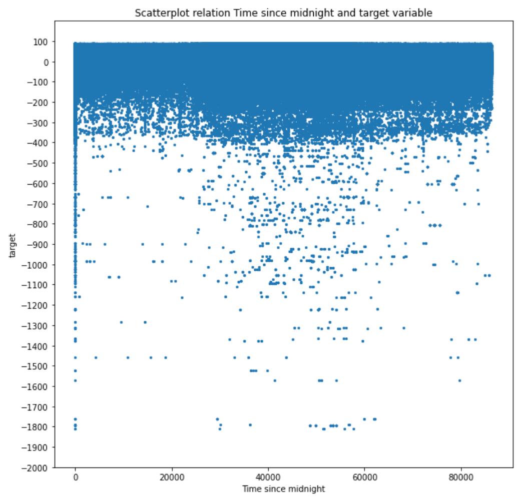


Figure 60: Scatterplot relation time since midnight and target variable

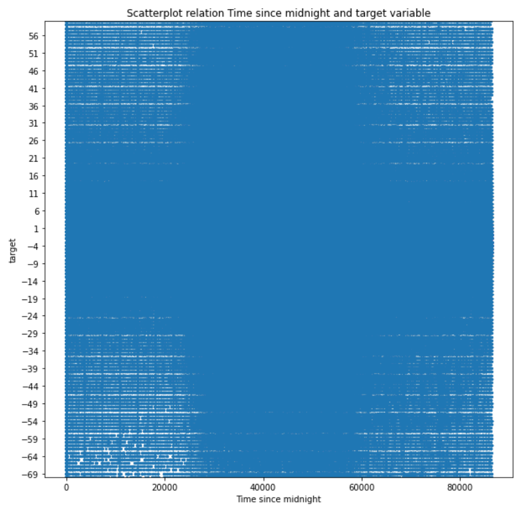


Figure 61: Scatterplot relation time since midnight and target variable - target range without outliers

Appendix B

This appendix contains the more detailed results related to analyses in Chapter 5.

Subsamples

Figure 62 shows the distribution of the target values for the different subsamples that have been created during the data preparation phase of the BPMP-PM.

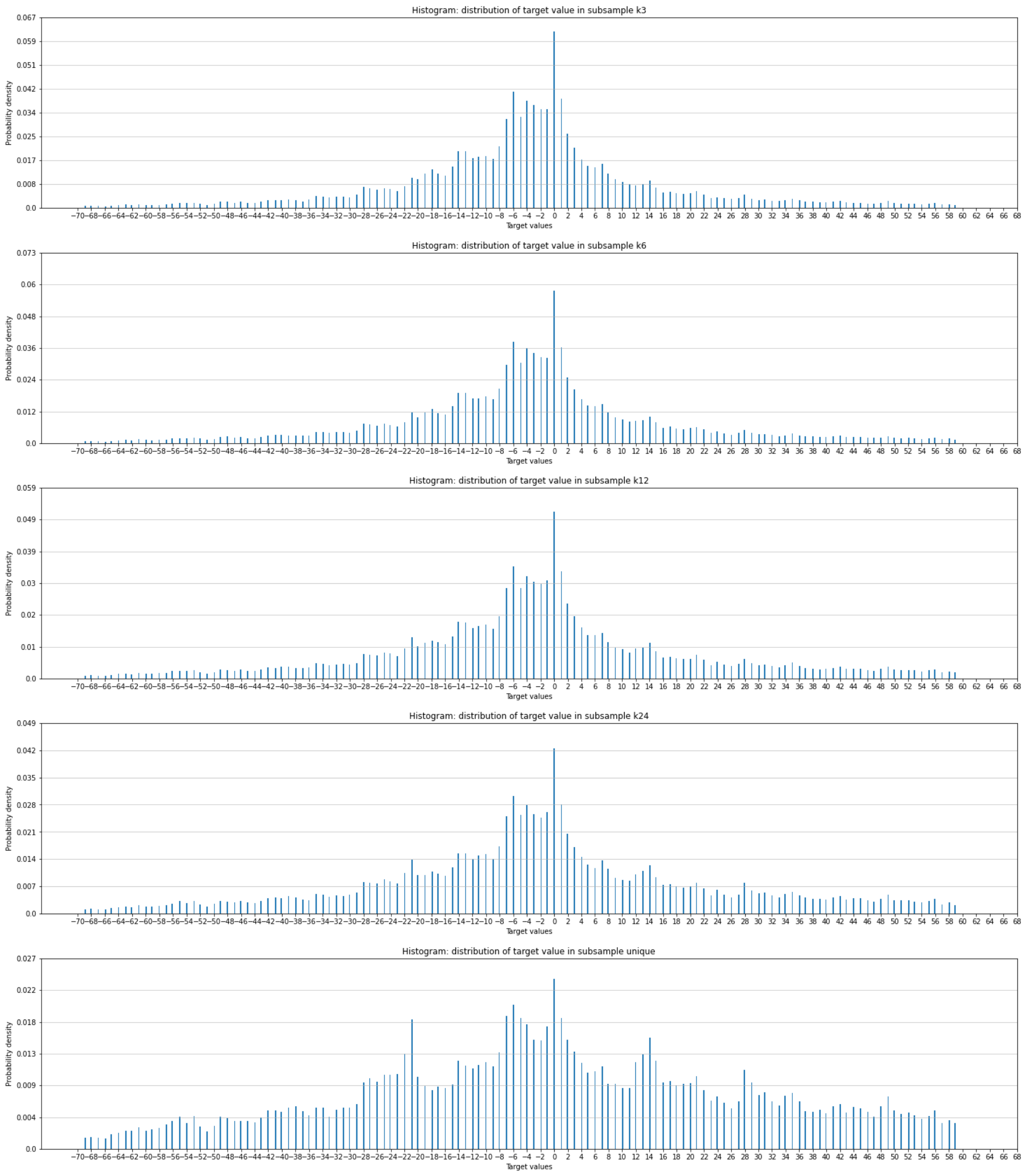


Figure 62: Distribution plot of the target variable for the subsamples

Moreover, Figures 63 - 67 depict the CDF comparison between the various subsamples and the total dataset.

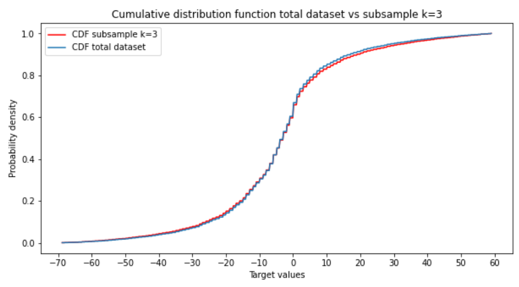


Figure 63: The CDF for the total dataset and subsample $k = 3$

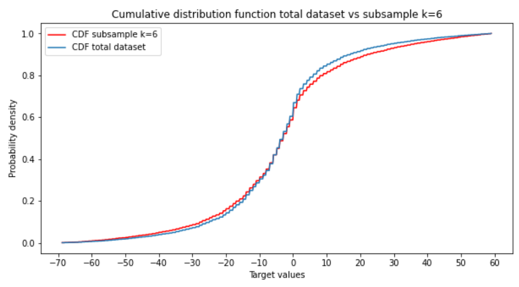


Figure 64: The CDF for the total dataset and subsample $k = 6$

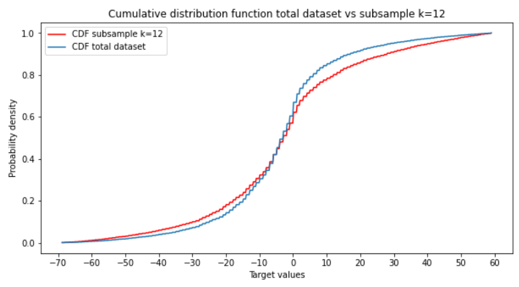


Figure 65: The CDF for the total dataset and subsample $k = 12$

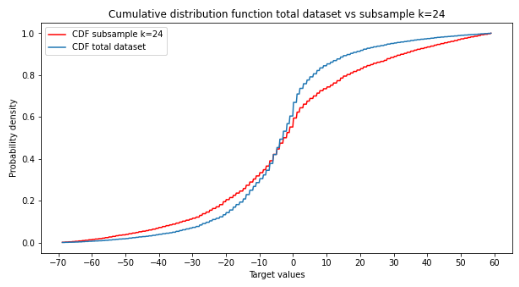


Figure 66: The CDF for the total dataset and subsample $k = 24$

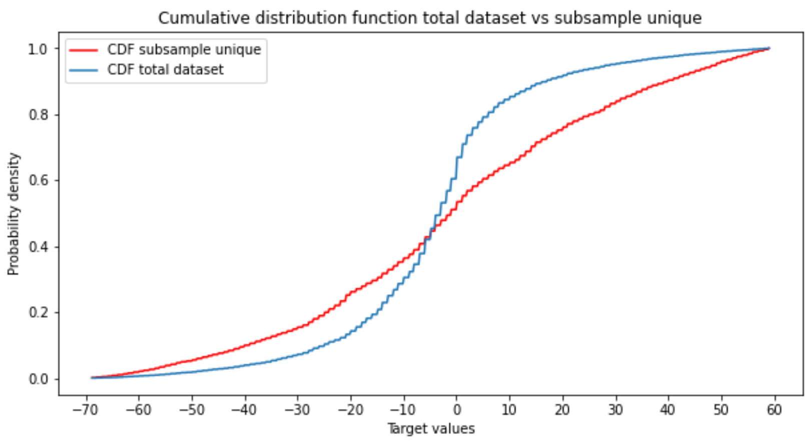


Figure 67: The CDF for the total dataset and subsample 'Unique'

NN Experiment Results

First of all, Table 49 shows the experimental decision factors and their ranges for the different experiments performed for the NNs.

Table 49: Experimental setup NNs

Decision factor	**Range**
Sample size	{subsample $k = 24$, subsample $k = 12$, subsample $k = 6$, subsample $k = 3$, total dataset}
Feature set	{*main feature set:* activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number; *aggregate feature set:* activity, time since case start, mean time since last event, mean time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number; *top (smaller) feature set:* time since case start, time since last event, time since midnight, BU, OTIF of last year, plant, PO lines, days between PO creation and contractual delivery date, material/part number; *smaller feature set 2:* time since case start, time since last event, time since midnight, OTIF of last year, PO lines, days between PO creation and contractual delivery date ; *smaller feature set 3:* time since case start, time since last event, time since midnight, OTIF of last year, PO lines, days between PO creation and contractual delivery date, activity, weekday; *CNN features (Pasquadibisceglie et al., 2020)*: time since case start, prefix length, mean time between events, median time between events, minimum time between events, maximum time between events, frequency of occurring activity pairs, number of occurrences for each activity value in the prefix; *BIG-DGCNN features (Chiorrini et al., 2022):* activity, time of the day, time of the week, time since last event; *Chiorrini et al. (2022) time features and own non-time features:* time of the day, time of the week, time since last event, activity, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number}

Parameters	number of neurons in hidden layers: {64, 100, 128, 300}
	learning rate: {0.01, 0.001, 0.0001}
	early stopping patience: {7, 10, 15}
	batch size: {1, 16, 32, 64, 256}
	number of training epochs: {30, 100}
	dropout: {0.1, 0.2}

Type of cases	{'too late' cases, 'normal' cases, all cases}

Feature encoding	categorical features: {one-hot, one-hot frequency based, labeling with integers and subsequently embedded}
	numerical features: {log-scaling, log-scaling and min-max normalization}
	target: {as-is, log-scaling}

Sequence encoding	{prefix padding, aggregation encoding}
Resampling method	{none, random over- and undersampling}

Next, Tables 50 - 53 summarize the experiments that have been executed for the different NN architectures and their results.

MLP

Table 50: MLP experiments summary

Subsample	Features	Feature encoding	Sequence encoding	Dataset modifications	Parameter settings	MAE (days)	training time (hours)
Subsample 'Unique'	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	learning rate of 0.0001, 100 neurons, 100 epochs, 1 run, early stopping patience of 15, batch size of 64 for train and test, 16 for valid	17.458	16.193 (100 epochs)
Subsample $k = 24$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	learning rate of 0.0001, 100 neurons, 100 epochs, 1 run, early stopping patience of 7, batch size of 64 for train and test, 16 for valid	13.786	11.253 (50 epochs)
Subsample $k = 12$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	learning rate of 0.0001, 100 neurons, 100 epochs, 1 run, early stopping patience of 15, batch size of 64 for train and test, 16 for valid	11.045	37.304 (100 epochs)
Subsample $k = 12$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: labeling with integers, subsequently embedded continuous: log-scaling target: as-is	*embedding matrix.* Separate prefix padded feature vectors for categorical features of shape 1 x maximum trace length, which are transformed into embedding matrix of shape unique feature values x embedding space. One prefix padded feature vector for all numerical features of shape number of features x maximum trace length. Those vectors are concatenated. The embeddings are learned simultaneously with the NN.	-	100 epochs, patience of 7, 1 run, 0.0001 learning rate, 32 batch size for training and validation, 100 neurons.	16.614	22.782
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	10.217	18.385 (37 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	random oversampling and undersampling applied.	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	10.010	22.895 (47 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: **log-scaling**	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	19.773	38.902 (80 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	Only 'too late' cases included, where target $\in (0, 59]$	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	9.875	12.894 (77 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	Only 'normal' cases included, where target $\in (-20, 20)$	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	5.024	41.723 (100 epochs)
Subsample $k = 6$	smaller feature set: time since case start, time since last event, time since midnight, BU, OTIF of last year, plant, PO lines, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	10.696	35.000 (76 epochs)

GCN

Table 51: GCN experiments summary

Subsample	Features	Feature encoding	Sequence encoding	Dataset modifications	Parameter settings	MAE (days)	training time (hours)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	13.096	7.119 (46 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: **log-scaling**	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	18.527	11.078 (73 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	Only 'too late' cases included, where target $\in (0, 59]$	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	11.104	3.837 (54 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	Only 'normal' cases included, where target $\in (-20, 20)$	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	6.053	4.676 (36 epochs)
Subsample $k = 6$	smaller feature set: time since case start, time since last event, time since midnight, BU, OTIF of last year, plant, PO lines, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape number of unique activities x number of features, for each prefix with length ≥ 1	-	100 neurons in hidden layers, batch size 64 for training, 16 for validation, and 1 for testing. Learning rate of 0.0001, early stopping patience of 10, 100 training epochs, 1 training run.	13.168	7.650 (38 epochs)

LSTM

Table 52: LSTM experiments summary

Subsample	Features	Feature encoding	Sequence encoding	Dataset modifications	Parameter settings	MAE (days)	training time (hours)
Subsample $k = 12$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	-	100 epochs, 1 run, 100 neurons, 64 training and validation batch size, 0.2 dropout, 0.001 learning rate, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	15.863	10 (100 epochs)
Subsample $k = 12$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: labeling with integers, subsequently embedded continuous: log-scaling target: as-is	*embedding matrix.* Separate prefix padded feature vectors for categorical features of shape 1 x maximum trace length, which are transformed into embedding matrix of shape unique feature values x embedding space. One prefix padded feature vector for all numerical features of shape number of features x maximum trace length. Those vectors are concatenated. The embeddings are learned simultaneously with the NN.	-	100 epochs, 1 run, 100 neurons, batch size of 64 for training and validation, 0.2 dropout, 0.001 learning rate, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	15.857	14.659 (100 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	-	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	13.642	0.918 (13 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	-	300 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	13.413	2.583 (12 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: **log-scaling**	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	-	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	13.651	0.771 (11 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	Only 'normal' cases included, where target $\in (-20, 20)$	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	6.062	0.607 (12 epochs)
Subsample $k = 6$	smaller feature set: time since case start, time since last event, time since midnight, BU, OTIF of last year, plant, PO lines, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	-	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	30.808	0.860 (11 epochs)
Subsample $k = 6$	smaller feature set 2: time since case start, time since last event, time since midnight, OTIF of last year, PO lines, days between PO creation and contractual delivery date.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	-	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	28.189	1.032 (13 epochs)
Subsample $k = 6$	smaller feature set 3: time since case start, time since last event, time since midnight, OTIF of last year, PO lines, days between PO creation and contractual delivery date, activity, weekday.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	-	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	30.482	1.119 (14 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot continuous: log-scaling target: as-is	*prefixes padded.* feature vector with shape maximum case length x number of features, for each prefix starting with length $\geq$ 1. Zero padding for where information is lacking.	Divided target variable up into 8 equal classes. Undersampling of majority class and subsequently oversampling of the two extreme classes (extreme early, extreme late)	100 neurons in hidden layers, batch size 64 for training and validation, and 1 for testing. Learning rate of 0.001, 100 epochs, 1 run, early stopping patience of 10, 1 training run, 0.2 dropout, 0.9 & 0.999 beta1 & beta2, 1e-08 epsilon, 0.004 scheduled decay, 3 clip value.	13.541	1.248 (16 epochs)

CNN

Table 53: CNN experiments summary

Subsample	Features	Feature encoding	Sequence encoding	Dataset modifications	Parameter settings	MAE (days)	training time (hours)
Subsample $k = 6$	features of Pasquadibisceglie et al. (2020): time since case start, prefix length, mean time between events, median time between events, minimum time between events, maximum time between events, frequency of occurring activity pairs, number of occurrences for each activity value in the prefix.	categorical: frequency-based continuous: min-max normalization target: as-is.	Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features. The features for which the prefix has no value, are filled with zeros.	-	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	12.797	10.532 (28 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot frequency based (dummies) continuous: log-scaling target: as-is	*Aggregation encoding.* Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features.	-	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	8.599	4.493 (27 epochs)
Subsample $k = 6$	activity, time since case start, **mean time since last event, mean time since midnight**, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot frequency based (dummies) continuous: log-scaling target: as-is	*Aggregation encoding.* Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features.	-	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	8.796	3.884 (24 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot frequency based (dummies) continuous: log-scaling target: **log-scaling**	*Aggregation encoding.* Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features.	-	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	8.954	5.218 (32 epochs)
Subsample $k = 6$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot frequency based (dummies) continuous: log-scaling target: as-is	*Aggregation encoding.* Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features.	Only 'normal' cases included, where target $\in (-20, 20)$	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	4.348	2.468 (21 epochs)
Subsample $k = 6$	smaller feature set: time since case start, time since last event, time since midnight, BU, OTIF of last year, plant, PO lines, days between PO creation and contractual delivery date, material/-part number.	categorical: one-hot frequency based (dummies) continuous: log-scaling target: as-is	*Aggregation encoding.* Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features.	-	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	9.228	1.984 (24 epochs)
Subsample $k = 3$	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot frequency based (dummies) continuous: log-scaling target: as-is	*Aggregation encoding.* Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features.	-	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	7.792	5.751 (20 epochs)
Total dataset	activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot frequency based (dummies) continuous: log-scaling target: as-is	*Aggregation encoding.* Feature vector for each prefix with length ≥ 1 has shape of number of 1 x number of features.	-	128 neurons in the dense hidden layers, batch size of 64 for training and validation, and 1 for testing. Learning rate of 0.0001, 100 epochs, 1 run, early stopping patience of 10.	7.239	28.103 (27 epochs)

BIG-DGCNN

Table 54: BIG-DGCNN experiments summary

Subsample	Features	Feature encoding	Sequence encoding	Dataset modifications	Parameter settings	MAE (days)	training time (hours)
Subsample $k = 12$	activity, time of the day, time of the week, time since last event	categorical: one-hot encoding numerical: min-max normalization target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm		learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 100 epochs, early stopping patience of 10.	15.811	34.5 (23 epochs)
Subsample $k = 6$	activity, time of the day, time of the week, time since last event	categorical: one-hot encoding numerical: min-max normalization target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	changed shape of target tensor in all dataloaders to the same shape as the predictions. this improves loss calculation.	learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 100 epochs, early stopping patience of 10.	12.202	56.563 (29 epochs)
Subsample $k = 6$	activity, time of the day, time of the week, time since last event	categorical: one-hot encoding numerical: min-max normalization target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation	learning rate of **0.01**, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 100 epochs, early stopping patience of 10.	12.337	23.225 (14 epochs)
Subsample $k = 6$	activity, time of the day, time of the week, time since last event	categorical: one-hot encoding numerical: min-max normalization target: **log-scaling**	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation	learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 100 epochs, early stopping patience of 10.	12.185	66.217 (29 epochs)
Subsample $k = 6$	activity, time of the day, time of the week, time since last event	categorical: one-hot encoding numerical: min-max normalization target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation. Only **'normal' cases** included, where target $\in (-20, 20)$	learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 30 epochs, early stopping patience of 10.	5.892	41.540 (30 epochs)
Subsample $k = 6$	Chiorrini et al. (2022) time features: time of the day, time of the week, time since last event, and **own non-time features**: activity, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot encoding numerical: min-max normalization for time features (Chiorrini et al., 2022), log-scaling for own features target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation.	learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 30 epochs, early stopping patience of 10.	6.627	70.554 (30 epochs)
Subsample $k = 6$	Chiorrini et al. (2022) time features: time of the day, time of the week, time since last event, and **own non-time features**: activity, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot encoding numerical: min-max normalization for time features (Chiorrini et al., 2022), log-scaling for own features target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation.	learning rate of 0.0001, 64 neurons, **batch size of 256 for training** and 64 for validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 30 epochs, early stopping patience of 10.	6.742	10.853 (30 epochs)
Subsample $k = 6$	Chiorrini et al. (2022) time features: time of the day, time of the week, time since last event, and **own non-time features**: activity, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot encoding numerical: min-max normalization for time features (Chiorrini et al., 2022), log-scaling for own features target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation.	learning rate of 0.0001, 64 neurons, **batch size of 256 for training** and 64 for validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, **100 epochs**, early stopping patience of 10.	6.644	36.621 (71 epochs)
Subsample $k = 6$	Chiorrini et al. (2022) time features: time of the day, time of the week, time since last event, and **own non-time features**: activity, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot encoding numerical: min-max normalization for time features (Chiorrini et al., 2022), log-scaling for own features target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation. Only **'normal' cases** included, where target $\in (-20, 20)$	learning rate of 0.0001, 64 neurons, batch size of 64 for training and validation, batch size of 1 for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 30 epochs, early stopping patience of 10.	3.275	43.434 (30 epochs)
Total dataset	Chiorrini et al. (2022) time features: time of the day, time of the week, time since last event, and **own non-time features**: activity, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number.	categorical: one-hot encoding numerical: min-max normalization for time features (Chiorrini et al., 2022), log-scaling for own features target: as-is	Instance graph for each prefix with length ≥ 3, created with the BIG-algorithm	improved loss calculation.	learning rate of 0.0001, 64 neurons, **batch size of 256 for training and 256 for validation**, batch size of half the size of the test set for testing, k = 15, dropout=0.1, number of layers = 5, 1 run, 30 epochs, early stopping patience of 10.	5.584	87.536 (30 epochs)

Best NNs
Figures 68 and 69 visualize the distribution of absolute error values for the CNN and BIG-DGCNN by means of a boxplot.

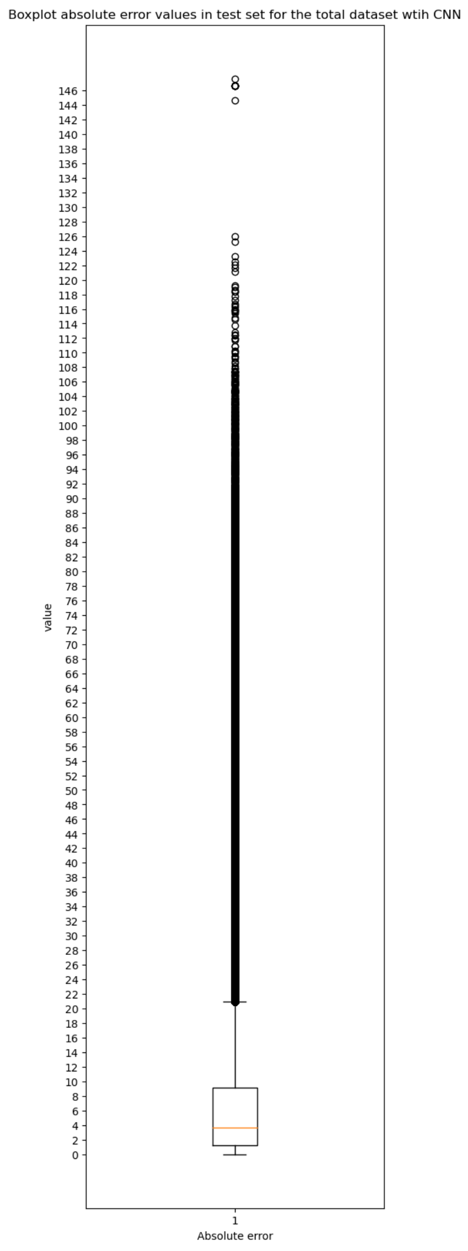


Figure 68: Boxplot of absolute error values for the CNN using the test split of the total dataset

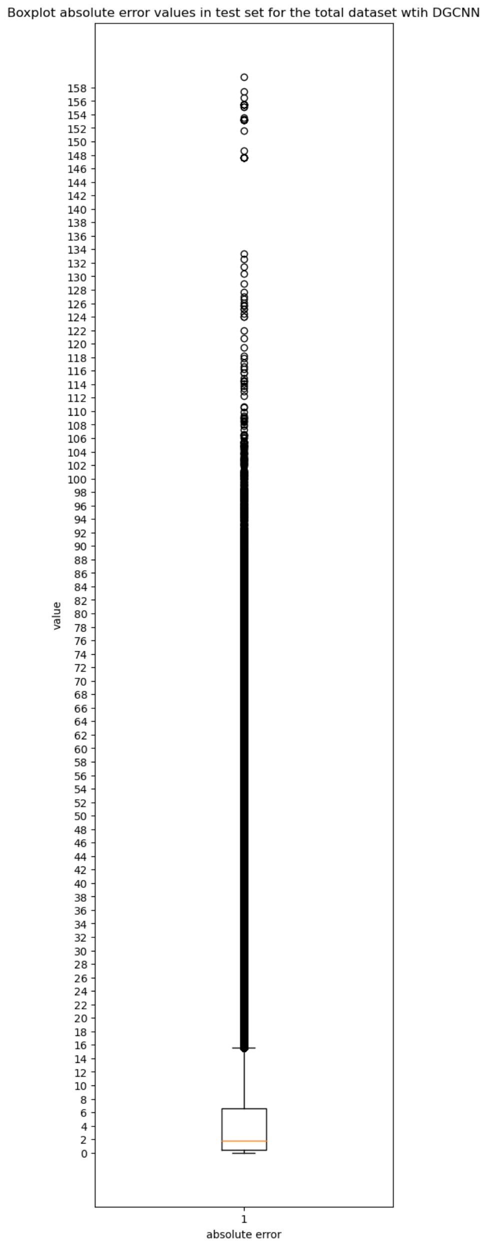


Figure 69: Boxplot of absolute error values for the BIG-DGCNN using the test split of subsample $k = 6$

Random Forest Experiment Results

First of all, Table 55 shows the experimental decision factors and ranges for the experiments performed for the various RFs.

Table 55: Experimental setup RFs

Decision factor	**Range**
Classification bins	{3 equal bins $\rightarrow$ 0: $(-69.001, -8.0]$ 1: $(-8.0, 0.0]$ 2: $(0.0, 59.0]$, 2 bins $\rightarrow$ 0: $(-69.001, 0.0]$ 1: $(0.0, 59.0]$, 3 domain-knowledge based bins $\rightarrow$ 0: $(-69.001, -20.0]$ 1: $(-20.0, 20.0)$ 2: $[20.0, 59.0]$, 3 domain-knowledge based bins with smaller middle bin $\rightarrow$ 0: $(-69.001, -9.0]$ 1: $(-9.0, 20.0)$ 2: $[20.0, 59.0]$}
Feature set	{*main feature set:* activity, time since case start, time since last event, time since midnight, weekday, BU, OTIF of last year, number of times delivery date has been changed, plant, PO lines, item, vendor, days between PO creation and contractual delivery date, material/part number; *top (smaller) feature set:* time since case start, time since last event, time since midnight, BU, OTIF of last year, plant, PO lines, days between PO creation and contractual delivery date, material/part number}
RF type	{Random Forest Classifier, Random Forest Classifier with random oversampling of the minority classes to contain $1/3^{rd}$ of the train set and random undersampling of the majority class to also contain $1/3^{rd}$ of the train set, Random Forest Classifier with random oversampling of the minority classes to 60% of the majority class, Random Forest Classifier with class_weight parameter on "balanced_subsample", Balanced Random Forest Classifier}

HP-optimization	trials: {5, 25}
	objective: {maximize mean validation accuracy, maximize mean validation f1 micro score, maximize mean validation f1 macro score, maximize composite loss which is the product of the mean validation accuracy, mean validation micro precision, and mean validation micro recall}
	parameters to optimize: {(max_depth, nr_estimators, max_features), (max_depth, nr_estimators, max_features, min_samples_split, min_samples_leaf)}

Cross-validation	{none, 5-fold cross-validation, 5-fold stratified cross-validation, 5-fold stratified cross-validation with over- and undersampling during each fold}

Next, Table 56 shows the results for the experiments performed for the Random Forest.

Table 56: Random Forest experiments summary

Subsample	Bins	RF type	Hyperoptimization	Cross-valdation	Parameter result	Performance measures	training time (hours)

Subsample
$k = 12$
	3 equal bins →
	0: $(-69.001, -8.0]$
	1: $(-8.0, 0.0]$
	2: $(0.0, 59.0]$
		Random Forest Classifier	25 trials, objective: maximize valdition accuracy, uniform ranges:
			max_depth=(10, 40)
			nr_estimators=(300, 600)
			max_features=(20, 71)
				no cross-validation	max_depth = 19
					nr_estimators = 350
					max_features = 23
						F1-macro =**0.685**
						precision = {0 : 0.708, 1 : 0.647, 2 : 0.714}
						recall= {0 : 0.818, 1 : 0.566, 2 : 0.673}
						F1 = {0 : 0.759, 1 : 0.604, 2 : 0.693}
							3.612

Subsample
$k = 12$
	2 bins →
	0: $(-69.001, 0.0]$
	1: $(0.0, 59.0]$
		Random Forest Classifier	25 trials, objective: maximize mean validation accuracy, uniform ranges:
			max_depth=(2, 40)
			nr_estimators=(100, 500)
			max_features=(10, 71)
			minsamplesleaf=(2, 10)
			minsamplessplit=(2, 10)
				5-fold cross-validation	max_depth = 18
					nr_estimators = 130
					max_features = 34
					minsamplesleaf = 9
					minsamplessplit = 6
						F1-macro = 0.664
						precision = {0 : 0.726, 1 : 0.643}
						recall= {0 : 0.836, 1 : 0.483}
						F1 = {0 : 0.777, 1 : 0.552}
							39.341

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier	5 trials, objective: maximize mean validation accuracy,
			ranges:
			max_depth=choice([10, 15, 20, 25, 30, 35, 40, 45])
			nr_estimators=uniform(300, 600)
			max_features=choice([20, 25, 30, 35, 40, 45, 50])
			minsamplesleaf=choice([2, 5, 8, 10])
			minsamplessplit=choice([5, 8, 10, 12, 15])
				5-fold cross-validation	max_depth = 15
					nr_estimators = 463
					max_features = 40
					minsamplesleaf = 10
					minsamplessplit = 12
						accuracy= 0.729
						F1-macro = 0.510
						precision = {0 : 0.560, 1 : 0.775, 2 : 0.607}
						recall= {0 : 0.636, 1 : 0.890, 2 : 0.058}
						F1 = {0 : 0.596, 1 : 0.828, 2 : 0.106}
							8.224

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier	5 trials, objective: maximize mean validation f1 micro score,
			ranges:
			max_depth=choice([10, 15, 20, 25, 30, 35, 40, 45])
			nr_estimators=uniform(300, 600)
			max_features=choice([20, 25, 30, 35, 40, 45, 50])
			minsamplesleaf=choice([2, 5, 8, 10])
			minsamplessplit=choice([5, 8, 10, 12, 15])
				5-fold cross-validation	max_depth = 45
					nr_estimators = 544
					max_features = 25
					minsamplesleaf = 8
					minsamplessplit = 15
						F1-micro = 0.730, F1-macro = 0.510
						precision = {0 : 0.561, 1 : 0.775, 2 : 0.622}
						recall= {0 : 0.636, 1 : 0.891, 2 : 0.057}
						F1 = {0 : 0.597, 1 : 0.829, 2 : 0.105}
							8.605

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier	5 trials, objective: maximize composite loss which is the product of mean validation accuracy, mean validation micro precision, and mean validation micro recall,
			ranges:
			max_depth=choice([10, 15, 20, 25, 30, 35, 40, 45])
			nr_estimators=uniform(300, 600)
			max_features=choice([20, 25, 30, 35, 40, 45, 50])
			minsamplesleaf=choice([2, 5, 8, 10])
			minsamplessplit=choice([5, 8, 10, 12, 15])
				5-fold cross-validation	max_depth = 45
					nr_estimators = 301
					max_features = 20
					minsamplesleaf = 8
					minsamplessplit = 12
						accuracy= 746,precision-micro = 0.746,recall-micro = 0.746, F1-macro = 0.508
						precision = {0 : 0.563, 1 : 0.774, 2 : 0.639}
						recall= {0 : 0.636, 1 : 0.893, 2 : 0.053}
						F1 = {0 : 0.597, 1 : 0.829, 2 : 0.099}
							8.009

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier	5 trials, objective: maximize mean f1-macro score,
			ranges:
			max_depth=choice([10, 15, 20, 25, 30, 35, 40, 45])
			nr_estimators=uniform(300, 600)
			max_features=choice([20, 25, 30, 35, 40, 45, 50])
			minsamplesleaf=choice([2, 5, 8, 10])
			minsamplessplit=choice([5, 8, 10, 12, 15])
				5-fold cross-validation	max_depth = 25
					nr_estimators = 495
					max_features = 40
					minsamplesleaf = 2
					minsamplessplit = 12
						F1-macro = 0.515
						precision = {0 : 0.551, 1 : 0.776, 2 : 0.475}
						recall= {0 : 0.634, 1 : 0.879, 2 : 0.076}
						F1 = {0 : 0.590, 1 : 0.824, 2 : 0.132}
							8.370

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier with random oversampling of the minority classes to contain $1/3^{rd}$ of the train set and random undersampling of the majority class (class 1) to also contain $1/3^{rd}$ of the train set	5 trials, objective: maximize mean validation f1-macro score,
			uniform ranges:
			max_depth=(15, 20)
			nr_estimators=(100, 200)
			max_features=(20, 50)
			minsamplesleaf=(10, 50)
			minsamplessplit=(10, 50)
				5-fold stratified cross-validation with over & undersampling during each fold	max_depth = 18
					nr_estimators = 124
					max_features = 33
					minsamplesleaf = 23
					minsamplessplit = 26
						F1-macro = **0.562**
						precision = {0 : 0.491, 1 : 0.884, 2 : 0.286}
						recall= {0 : 0.829, 1 : 0.585, 2 : 0.508}
						F1 = {0 : 0.617, 1 : 0.704, 2 : 0.366}
							48.931

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier with random oversampling of the minority classes to contain $1/3^{rd}$ of the train set and random undersampling of the majority class (class 1) to also contain $1/3^{rd}$ of the train set. Using the **smaller feature set**.	5 trials, objective: maximize mean validation f1-macro score,
			uniform ranges:
			max_depth=(15, 20)
			nr_estimators=(100, 200)
			max_features=(20, 50)
			minsamplesleaf=(10, 50)
			minsamplessplit=(10, 50)
				5-fold stratified cross-validation with over & undersampling during each fold	max_depth = 18
					nr_estimators = 103
					max_features = 39
					minsamplesleaf = 10
					minsamplessplit = 13
						F1-macro = 0.553
						precision = {0 : 0.496, 1 : 0.863, 2 : 0.269}
						recall= {0 : 0.761, 1 : 0.617, 2 : 0.460}
						F1 = {0 : 0.600, 1 : 0.719, 2 : 0.339}
							48.761

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier with random oversampling of the minority classes to 60% of the majority class (class 1)	5 trials, objective: maximize mean validation f1-macro score,
			uniform ranges:
			max_depth=(15, 20)
			nr_estimators=(100, 200)
			max_features=(20, 50)
			minsamplesleaf=(10, 50)
			minsamplessplit=(10, 50)
				5-fold stratified cross-validation with oversampling during each fold	max_depth = 18
					nr_estimators = 168
					max_features = 31
					minsamplesleaf = 36
					minsamplessplit = 44
						F1-macro = **0.582**
						precision = {0 : 0.505, 1 : 0.844, 2 : 0.377}
						recall= {0 : 0.802, 1 : 0.752, 2 : 0.294}
						F1 = {0 : 0.620, 1 : 0.795, 2 : 0.331}
							48.524

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier with class_weight parameter on "balanced_subsample"	5 trials, objective: maximize mean validation f1-macro score,
			uniform ranges:
			max_depth=(15, 50)
			nr_estimators=(100, 600)
			max_features=(20, 50)
			minsamplesleaf=(10, 50)
			minsamplessplit=(10, 50)
				5-fold stratified cross-validation	max_depth = 46
					nr_estimators = 504
					max_features = 20
					minsamplesleaf = 17
					minsamplessplit = 19
						F1-macro = **0.576**
						precision = {0 : 0.489, 1 : 0.875, 2 : 0.317}
						recall = {0 : 0.838, 1 : 0.645, 2 : 0.435}
						F1 = {0 : 0.618, 1 : 0.742, 2 : 0.367}
							50

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier with class_weight parameter on "balanced_subsample" using the smaller feature set	5 trials, objective: maximize mean validation f1-macro score,
			uniform ranges:
			max_depth=(15, 50)
			nr_estimators=(100, 600)
			max_features=(20, 50)
			minsamplesleaf=(10, 50)
			minsamplessplit=(10, 50)
				5-fold stratified cross-validation	max_depth = 27
					nr_estimators = 266
					max_features = 43
					minsamplesleaf = 15
					minsamplessplit = 40
						F1-macro = 0.569
						precision = {0 : 0.502, 1 : 0.867, 2 : 0.293}
						recall= {0 : 0.787, 1 : 0.647, 2 : 0.444}
						F1 = {0 : 0.613, 1 : 0.741, 2 : 0.353}
							48.758

Subsample
$k = 12$
	3 domain-knowledge based bins→
	0 :(-69.001, -20.0]
	1 :(-20.0, 20.0)
	2 :[20.0, 59.0]
		Balanced Random Forest Classifier	5 trials, objective: maximize mean validation f1-macro score,
			uniform ranges:
			max_depth=(15, 50)
			nr_estimators=(100, 600)
			max_features=(20, 50)
			minsamplesleaf=(10, 50)
			minsamplessplit=(10, 50)
				5-fold stratified cross-validation	max_depth =44
					nr_estimators =482
					max_features =34
					minsamplesleaf = 11
					minsamplessplit = 27
						F1-macro = **0.558**
						precision = {0 : 0.491, 1 : 0.891, 2 : 0.279}
						recall = {0 : 0.829, 1 : 0.562, 2 : 0.539}
						F1 = {0 : 0.617, 1 : 0.689, 2 : 0.368}
							48.917

Subsample
$k = 12$
	3 domain-knowledge based bins with smaller middle bin→
	0 :(-69.001, -9.0]
	1 :(-9.0, 20.0)
	2 :[20.0, 59.0]
		Random Forest Classifier	5 trials, objective: maximize mean validation accuracy,
			ranges:
			max_depth=choice([10, 15, 20, 25, 30, 35, 40, 45])
			nr_estimators=uniform(300, 600)
			max_features=choice([20, 25, 30, 35, 40, 45, 50])
			minsamplesleaf=choice([2, 5, 8, 10])
			minsamplessplit=choice([5, 8, 10, 12, 15])
				5-fold cross-validation	max_depth =10
					nr_estimators =313
					max_features =40
					minsamplesleaf = 10
					minsamplessplit = 12
						accuracy= 0.682,F1-macro = 0.512
						precision = {0 : 0.624, 1 : 0.725, 2 : 0.687}
						recall= {0 : 0.794, 1 : 0.779, 2 : 0.045}
						F1 = {0 : 0.699, 1 : 0.751, 2 : 0.084}
							9.740

Appendix C

Detailed Predictive Process Monitoring Steps

This section provides the details regarding the method implementation step within the predictive process monitoring part of this thesis.

Method Implementation
Within this section all the modifications are explained which have been performed to make the five NNs work for the prediction task as well as the selected input data.

MLP

First of all, the function creating the prefix-based feature vectors was modified. The computation of the time features (time since case start, time since last event, time since midnight, and weekday) could be removed as this was already done within Celonis from which the features were directly exported. Moreover, the if-statement determining the start of a new case was changed. Originally, the start of a case was defined by the time since last event being zero for a prefix. However, this would not work for the cases within our P2P dataset, as time since last event could be zero multiple times for events in a case, for example when events are performed in parallel. Therefore, this was changed to an if-statement defining the start of a case by the rule that it had to be the first occurence of a prefix with time since case start being zero.
Furthermore, the function creating the target vector was changed such that it took the given days between contractual and actual delivery date instead of the timestamp of next event.
Moreover, the shuffle parameter in the *train_test_split*-function was set to False for the split performed between the train and validation set. Otherwise, the prefixes from the same case could easily end up in both the train and validation set leading to data leakage.
Like explained in Section 5.1.1, Python code was added to check whether all datasets contained all prefixes for their cases.
Additionally, a function to save the case IDs was built, returning a list of case IDs such that the order corresponded to that of the feature vector and target vector. To be able to return the case IDs together with the prefixes and targets, the case IDs were included in the event log data object and added to the get_item function from the object.
To prevent overfitting, early stopping was included in the training and validation process. First, the variables *triggertimes* and *patience* were created. Patience was given a predetermined value indicating a threshold for the number of epochs that the validation loss was allowed to be unimproved compared to the best loss value found so far. The variable triggertimes was initiated at zero and increased with one in case the average validation loss for the data evaluated in the epoch was not below the best validation loss so far. When the value of triggertimes reached the value set for the patience, the training process was stopped.
Furthermore, the argument set_to_none in the *optimizer.zero_grad*-function was set to True. This sets the gradients to value None instead of zero. This reduces memory usage and can slightly improve performance (PyTorch Contributors, 2022b).
In the end, the average loss did not have to be divided by 86400 because the unit for the target values was already days instead of seconds.
Lastly, a variable called *trace_info* was added to save the case ID, prefix input, target value, prediction value, and loss for each prefix for the validation and test set.

GCN

The particular adaptation that had to be performed for the GCN was related to the adjacency matrix. That is, the function creating the process graph was changed by adding self loops to the adjacency matrix. Accordingly, the identity matrix was added to the adjacency matrix, $\tilde{A} = A + I$ (Chiorrini et al., 2022). Moreover the degree matrix (D) calculation was changed to $\tilde{D} = \sum_j \tilde{A}_{ij}$, where the degree matrix was computed using the adjusted adjacency matrix (Chiorrini et al., 2022). This was necessary due to the fact that the traces in the P2P log resulted in directed graphs. Apparently, there were no outgoing arcs for the 'record goods receipt' event due to it being the last event for all traces. Since the degree matrix shows the number of edges to which a node in a graph is connected, a row of zeros appeared in the degree matrix for this specific event. This generated an error in the calculation of the propagation rule from (Venugopal et al., 2021): $f(X, A, W) = \sigma(D^{-1/2}AD^{-1/2}XW)$.

LSTM

First of all, a couple of changes had to be made regarding the model, since the original model had a classification prediction task while the predictions for Company X are of a regression type. Therefore, the last Dense layer of model was adjusted such that it was suitable for regression. Hence, the number of output units in this layer was changed from the number of unique activities to a value of 1. Moreover, the Softmax activation being performed in the final layer was changed into no activation being performed. Furthermore, the loss function had to be altered by replacing categorical cross entropy with the mean absolute error (L1-loss). Besides, the metric to monitor was changed from categorical accuracy to mean absolute error. Additionally, the loss used to monitor model performance and base the choice of the best model on, was turned into the validation loss, instead of the prior training loss. Next, the predicted values were not retrieved using an Argmax-function, since regression was performed rather than classification. Unlike for classification, the model output values are directly the prediction values that were aimed for. With classification a probability for each possible output class is given as prediction and therefore an Argmax-function would be used to retrieve the output class for which the highest probability is given (Brownlee, 2020).
Furthermore, several other modifications were executed to achieve a model suitable for the prediction task and P2P dataset.
Firstly, an important change was made to the function creating the prefix feature (input) vectors (*build_with_padding*-function). Within this function, the dataset was grouped on case ID with a *pandas groupby*-function. In this groupby function the argument 'sort' was changed to the value False such that the order of cases as in the event log remained. This way, the cases were kept in chronological order, which is important as explained in Section 5.1.1.
Next, the defining the target values for all the prefixes was removed. This function was superfluous since the target values were already included in the dataset as exported from Celonis. The target values were engineered in Celonis prior to the data export (see Section 4.3).
Since, the original model had a next activity prediction task, the data for the last event in the case was removed. The next event did not have to be predicted when the case was already finished. However, this was not the case for our prediction task. Thus, the data for the last event of a case was not removed.
Additionally, the time features of Tax et al. (2017) were replaced by the self-defined time features, like was done for the MLP.
Moreover, an additional data split was performed on the training set to obtain a validation set. In the original code only a train and test set was generated by performing a 67-33% split. Besides, a function was added to obtain a list storing the case IDs for the prefixes. Then, when performing the split, this was done for the feature vector for all the prefixes, the target value vector, and list containing the case IDs. This was necessary to trace back the results for the prefixes to their corresponding cases. Then, like explained in Section 5.1.1 code was added to check if all datasets contained all prefixes for their cases.
Finally, early stopping was added to the training process. This was achieved by creating an object of the EarlyStopping class of Tensorflow (Tensorflow, 2022a). The monitoring metric validation loss and patience were provided as arguments to the class. Subsequently, the early stopping object was added to the list of callbacks in the *model.fit*-function from Tensorflow (Tensorflow, 2022c). This assured breaking off the training process when the monitored metric stopped improving for the number of epochs equal to the patience value.

CNN

First of all, the feature vector generation process was revised. In the final implementation of the model only the self-selected features were used (see Section 4.3.4). Therefore, the calculations for the features as in (Pasquadibisceglie et al., 2020) were replaced by two functions that created the prefix feature vectors by means of aggregate encoding. This method has already been explained in Section 5.1.1 and an example vector is shown in Table 29. The one-hot vectors for features activity and weekday were transformed into frequency one-hot vectors. Hence, the cases in the event log were grouped on case ID using a Pandas groupby-function and calculating the cumulative sum over the dummy columns (NumFOCUS, 2022b). Then, lists were created for each prefix in which the transformed activity and weekday features were added to the other features. These lists for all prefixes were again added to one list that served as model input.
Besides, just like for the other models, a function was included in the code to get the caseIDs corresponding to the prefixes in the dataset.
Furthermore, the code for encoding the target variable was removed. The target values were kept as-is, or were log-scaled within the experiments.
Moreover, within the image generation file, an extra function was added defining the index for the splitting point in the train dataset to turn it into a train and validation dataset. Like with the LSTM, originally only a train and test dataset were created.
Moreover, multiple changes were implemented regarding the CNN model and model training proces. Alike for the LSTM, the final Dense layer was changed by setting the output unit to a value of 1, because only one prediction value is needed as outcome. Moreover, the activation function was changed from Softmax to no activation. Futhermore, the categorical cross entropy loss function was replaced the mean absolute error, i.e. L1-loss. Besides, the accuracy was substituted by the mean absolute error as metric to monitor. In addition, the validation split argument in the *model.fit*-function was revised (Tensorflow, 2022c). from into " validation data = image val, y val. Apparently, a validation split was performed in the original code, however after image creation, by using the argument "validation split = 0.2". But, like explained above, it was chosen to define the validation split with an own function to ensure that the split was performed for entire cases, and no prefixes from a single case would be diffused over multiple partitions of the dataset.
Furthermore, a ModelCheckpoint object from Tensorflow was added to the list of callbacks in the *model.fit*-function, so the best model in terms of validation loss would automatically be saved (Tensorflow, 2022b,c).
The original hyper-parameter optimization was removed due to a huge increase in computational time when implementing it. Performing two model evaluations already took around 40 hours.
Finally, again the predicted values were not retrieved using an Argmax-function, since regression is performed instead of classification. Instead, the prediction output from the model is directly the value that is aimed for.

BIG-DGCNN

These changes performed for the BIG-DGCNN have been classified into changes related to the graph enrichment, the graph creation, and the model. All steps are explained in detail in the upcoming paragraphs.

Graph Enrichment
In this part of the approach of Chiorrini et al. (2022) the earlier created instance graphs for the cases are enriched with extra features. These extra features are the time features as defined by Chiorrini et al. (2022). Instead using the self-selected time features like was done for the other NNs, the original features were used. This is because process structure is taken into account in the calculation of these time features, which is one of the advantages of the approach of Chiorrini et al. (2022). The enriched graph data is subsequently stored in a text file to be retrieved later on in the procedure and added as attributes to Networkx graph objects (NetworkX Developers, 2022).
One of the problems with the implementation that occurred was that the time feature engineering and graph enrichment procedure as designed in Chiorrini et al. (2022) was extremely time consum-

ing for the P2P dataset, even for the smaller subsamples. For example, when running the procedure for the subsample $k = 6$, it would run over 40 hours without any produced result. Therefore, the procedure was redesigned in Python. After redesigning the procedure the feature engineering and graph enrichment could be performed within 16 minutes for the subsample $k = 6$.
In general, the computational efficiency increase was accomplished by replacing most for-loops, lambda functions, list appending functions, and string concatenations using the + operator, with more efficient constructs. For example, for-loops take extremely long when looping over a dataset like subsample $k = 6$ with around 1.7M lines. Multiple examples of how the graph feature enrichment code was adjusted to increase efficiency, are given in the next paragraph.

The main aim of the graph enrichment was to create a dataframe representing the graphs for all the cases, thus including the nodes with all their corresponding features as well as the edges. This dataframe was then eventually converted into a text file by writing each row of the dataframe as a line in the text file. The so-called graph dataframe was obtained by creating a dataframe in the structure of the instance graphs and by adding features step by step for the graph nodes. These features were added as an extra column to the dataframe.

In order to calculate the several time features, the start and finish time of an event needed to be available in the dataframe. The finish time was already included in the dataframe, since this equalled the time stamp of the event. Therefore, the start time of an event needed to be added as column to the dataframe. The start time value for a node was equal to the value for the finish time of the previous node, except for the first node of each node where start time of a node was equal to the finish time of itself. In the original code this start time was added to the dataframe as follows. The already existing graph dataframe was rebuilt row by row into a new graph dataframe, using an extensive for-loop. This loop checked all predecessors for an event within a case (hence a graph object was created for each case) and scanned them all to find the maximum previous event time. This information was added to a another dataframe, then appended to the graph dataframe, and repeated for each case. This time consuming method was replaced by the following construct. First, a copy of the already existing graph dataframe was created where all rows were downshifted by one row as a row with zeros was added at the top of the frame. By downshifting the rows, the finish time column of the shifted dataframe became the start time column as was needed for in the existing graph dataframe. Hence, the finish time column of the shifted dataframe was added as start time column to the already existing graph dataframe. Subsequently, a lambda function was applied on the start time column to ensure that the first node of each case got a start time value equal to the finish time value of that node.

Another example considers the way that the feature time of the week was normalized. The original code is explained in the pseudocode in Algorithm 3.

Algorithm 3: Pseudo-code of ORIGINAL normalizing time procedure

```
1  def get_normalized_date_of_week(date):
2      day_ofweek= datetime.datetime.weekday(date) ;                     ▷ Transform date into weekday
3      norm_timeofweek=(day_ofweek*24*60) + ((date.hour*60) + date.minute) ;    ▷ Transform weekday into
        minutes
4      norm_timeofweek=norm_timeofweek/10080 ;              ▷ Divide value by number of minutes in a week
5      return norm_timeofweek;
6  norm_time_list=[];
7  base = g_dataframe['finish'] ;                                         ▷ Column to normalize
8  for index in g_dataframe['finish'] do
9      norm_date=get_normalized_date_of_week(base[index]) ; ▷ Calculate normalized day of week value for an
        event
10     norm_time_list.append(norm_date) ;                                     ▷ Save value in a list
11 end
12 g_dataframe['norm_time']=pd.DataFrame(norm_time_list) ;          ▷ Assign the normalized values to a new
    feature column in the graph dataframe
```

This code was redesigned as shown in the pseudocode in Algorithm 4. Instead of looping over all the elements in the series and performing the normalizing operation sequentially for each single element, the normalization operation was changed to be performed simultaneously for all elements in the series.

Algorithm 4: Pseudo-code of REDESIGNED normalizing time procedure

```
1 g_dataframe['norm_time']=((g_dataframe['finish'].dt.dayofweek*24*60)+
    ((g_dataframe['finish'].dt.hour*60) + g_dataframe['finish'].dt.minute))/10080 ;   ▷ Do all computations in one
    go simultaneously for all values in the column and immediately assing the values to a new
    column
```

Many calculations regarding the other time features of Chiorrini et al. (2022) were changed using this same principle of performing operations simultaneously instead of sequentially.

The next example considers the procedure of converting the graph dataframe into one string object. The graph dataframe had to be transformed into one string object in order to save it in a text format. The original procedure is explained with the pseudocode in Algorithm 5.

Algorithm 5: Pseudo-code of the ORIGINAL graph dataframe to string conversion procedure

```
1 string_g_dataframe= ' ' ;                                          ▷ Initiate string variable
2 for index,row in graph dataframe do
3     for item in the graph dataframe columns do
4         string_g_dataframe+= string(row[item]) + ' ' ;   ▷ Concatenate the column values for one row into
            one string with a space between each column value and add to the final string variable
5     end
6     string_g_dataframe=string_g_dataframe.strip() ;   ▷ Removes spaces from both the beginning and end of
        the string
7     string_g_dataframe+= '/n' ;                      ▷ Makes sure a new line is started after this row
8 end
9 string_g_dataframe = string_g_dataframe.replace("nan", "");         ▷ Remove NaN (Not a Number) values
```

The pseudocode in Algorithm 6 shows the redesigned procedure. First the string object was created step by step, by adding sequentially for each row, each column value to a string object using the + -operator. This was changed by applying an aggregation function to the graph dataframe concatenating for each row the column values into one string. Subsequently, these strings for each row were transformed simultaneously into one string object with each string being a new line.

Algorithm 6: Pseudo-code of the REDESIGNED graph dataframe to string conversion procedure

```
1 Set data type for values in all columns of the graph dataframe to a string;
2 g_dataframe['string_value']= g_dataframe[all columns].T.agg(' '.join) ;   ▷ Creates a new column with as value
    a concatenated string of the column values with a space between each column value,
    simultaneously for each row
3 g_dataframe['string_value']=g_dataframe['string_value'].str.strip() ;   ▷ Does stripping of the string for each
    row simultaneously
4 string_g_dataframe= g_dataframe['string_value'].str.cat(sep='/n') ;   ▷ Transforms the column into one string
    value where each row in the column is a new line
5 string_g_dataframe+= '/n' ;                              ▷ Add final new line like in original code
6 string_g_dataframe = string_g_dataframe.replace("nan", "");          ▷ Remove NaN (Not a Number) values
```

Lastly, the target value had to be added to the graph dataframe as well so it was stored in the text file and could be retrieved for each case later on in the approach and added to the graph Networkx objects. This was not necessary in the original approach of Chiorrini et al. (2022), since the goal was next activity prediction. Therefore, the target values could be retrieved from the activity attribute already present in the graph dataframe.

The first step considered importing the case IDs and target values from the P2P-dataset. Since the cases in this dataframe do not contain the artificial start and end events the length of the cases differed compared to that of the graph dataframe, meaning the target column from the dataset could not immediately be added to the graph dataframe. A new array needed to be created for the target values where the target value appeared as much times as events in a case for each of the cases in the graph dataframe. The target graph addition procedure is explained in detail by the pseudocode in Algorithm 7.

Algorithm 7: Pseudo-code of target graph addition procedure

```
1  targetframe= Load dataframe containing case ID and target values;
2  targetframe=targetframe.groupby('caseID',sort=False,as_index=False)['target']; ▷ Create groupby object where
     cases are grouped and order of cases in targetframe is maintained
3  target_values=targetframe.agg('max')['target'] ; ▷ Take maximum target value for each case.  The value is
     the same for each event so taking the maximum is just a way to obtain one value for each case.
4  sizes= np.array(g_dataframe.groupby('caseID', sort=False, as_index=False.size()['size'] ;  ▷ Retrieve the length
     of each case in the graph dataframe and store in array.  The values in sizes and target_values
     are ordered similarly such that they belong to the same cases.
5  target_columnvalues=sum([[s] * n for s,n in zip(values,sizes)], [ ]) ; ▷ Creates a list where for each case the
     target value appears as often as the number of events in the case.
6  g_dataframe['target'] = [np.nan] * length(g_dataframe) ;     ▷ Intiate target column as a column with only
     NaN values
7  idxs=list(indexes where g_dataframe has a value for caseID) ; ▷ only the nodes have a value for caseID and
     for these indices the column target needs to have a value
8  Change target_columnvalues from array to a pandas series;
9  target_columnvalues.index=idxs ;    ▷ Set index values similar to the indices where the column target
     in the graph dataframe needs to have a value
10 g_dataframe['target']=target_columnvalues ;  ▷ The target values are assigned to the correct indices in
     the frame
```

Other own features were added to the graph dataframe in a similar fashion. However, a special not needs to be added on the categorical features. Instead of adding each dummy column separately to the graph dataframe, only one column for each categorical feature was added to the frame. The values for the column of the categorical feature were the dummy labels for which the node had value 1. Later in the actual graph creation these labels were transformed back into a one-hot encoding using the same mapping procedure as was originally performed for the activities. This saved computational time for the feature enrichment procedure.
The following paragraph describes the changes that have been performed to the graph creation procedure.

Graph Creation
The aim of this procedure was to turn the cases as stored in the enriched graph text file into actual NetworkX graph objects (NetworkX Developers, 2022). The procedure had to be adjusted due to having more features than used in the approach of Chiorrini et al. (2022) as well as another target. Therefore, an extra node attribute was added to the graphs created for the cases, for each self-selected attributes and the target. Moreover, an extra node attribute was added to the sub graphs corresponding to the prefixes, for each of the self-selected attributes. The target value assigned to the prefix (named SubGraph.graph['target_std'] in the code) was set equal to the target value that was given to the last event of the prefix in the corresponding case.
Furthermore, changes regarding the one-hot encoding of the categorical features had to be implemented. Since, more categorical features were added to the graphs besides activity, a text file containing all possible category values, had to be created for activity and all the self-selected categorical attributes. Next, the one-hot encoding dictionaries were then also created for all categorical attributes. The dictionaries were subsequently applied to the categorical feature values when adding them to the x feature vector of the prefix graphs, such that the features were added in a one-hot encoded format. The last paragraphs elaborates on the modifications that had to be applied to the model.

Model
First of all, the function performing the dataset split was changed. An extra split was performed on the training set to divide it into a train and validation set. The original approach of Chiorrini et al. (2022) only contained a train and test set. Furthermore, like for the other NNs code was added to check if all datasets contained all prefixes for their cases. Additionally, the part of the code was removed that selected an equal amount of prefixes for each target value to be in the various partitions of the total dataset.
Since a validation dataset was created, a dataloader for the validation set was created as well.
For the reason that regression is performed instead of classification, the criterion was changed from

cross entropy loss to mean absolute errror/L1 loss. On top of that, the last (linear) layer in the NN was modified such that it had only 1 output unit, and no activation was performed. Besides, like for the LSTM and CNN no Argmax-function needed to be applied to the model output to obtain the prediction values. In addition, no confusion matrix was generated to evaluate the prediction results, again because of performing regression instead of classification.
Then, alike the MLP and GCN the argument set_to_non in the *optimizer.zero_grad*-function was set to True, to reduce memory usage.
Finally, the most important change regarding the model was the following. The shape of the target tensor had to be adjusted such that it got the same shape as the prediction tensor. It was for example noted that the target tensor was a *torch tensor([23])* while the prediction tensor was a *torch tensor([23,1])*. Because the prediction tensor had a different shape than the target tensor, a problem arose with computing the loss. This resulted in getting the same prediction value for each input prefix.

Appendix D

Interview with the supply chain manager at Company X

Business context

- **What is your core business?**
  The main activities are identifying the ore body, mining, extracting, processing and selling. The activities are related to capital equipment, maintenance, and support.

- **How does your business network look like? What is the supply chain?**
  In the mining industry the supply chain looks quite different than for most companies. There are only a few customers, ordering large quantities of for example cole or platinum. On the other side, we need huge supplier base. Inbound supply chain becomes then quite important. The supply chain department needs to take care of getting the appropriate raw materials, spare parts, supporting goods and services. Examples of raw materials are fuel to go in the trucks, explosives, drills, and process chemicals. The spare parts are needed to maintain fleet and equipment.

- **Who are the competitors?**
  There are only a few competitors, around 5 to 10 global mining companies. Others are smaller and are not considered as competitors.

Purchase-to-Pay process

- **What is the happy flow of the process?**
  First of all, 40% of requests for purchasing is for items supporting maintenance operations, 15% from inventory replenishment and balance, and 45% comes from ad hoc request. Then, the happy flow consists of the following sequence of activities: (1) purchase request, (2) release purchase request, (3) create purchase order, (4) record goods receipt, (5) receive invoice, (6) clear invoice. These are the main activities and there are some other steps possible in between "create purchase order" and "record goods receipt".

- **Are there typical bottlenecks in the process?**
  The activity create purchase order is a bottleneck. We are trying to move to a more automatic creation of like 70-75%. But, now due to manual labor it takes a bit longer sometimes. Moreover, late deliveries.

- **What KPIs do you measure for this process?**
  Purchasing automation, PR to PO cycle time which is the time from release of purchase requestion to creating the PO, rework which includes all change activities, and OTIF which is the rate of items delivered on-time and in full.

Problem/Motivation

- **What is the underlying motivation for doing this project with me? What are the issues currently existing in the purchase to pay process?**
  Mainly, improving the OTIF receiving KPI. This allows for having ahead on time contact, and making alternative plans with respect to maintenance tasks.

- **Do you see typical/recurring reasons for why products are delivered too late?**
  There are no reasons allocated to that yet. However, it is seen that something is late because either there is an actual late delivery by items of suppliers, or it is late registered in the system. It can be for example that a physical item was there on Monday, but was only put on received in the system on Thursday. The problem with the last option is that items can be issued only if they are registered in the system. Another interesting thing is that there is data in Celonis about when the invoice is created by supplier. If it is assumed that this date is the same as the date the item leaves the facility then you know when it is sent.

- **What value will/should it give you and Company X?**
  Improve on time receiving of items. Which improves maintenance completion, which in turn improves machine availability, which then improves the number of productive hours and volume of saleable products.

Predictive model

- **What should be predicted exactly? One thing or multiple things? (e.g. date of delivery, classification of too late or not?, etc.)**
  The expected delivery date should be predicted. Not for purchase order line but actually for scheduled lines. This means there is a purchase order line for item 1 with quantity 100, that there can be 20 lines for delivery of quantity 5 of this item. However, it is not that important for Company X as in general we only have single scheduled line, one for each purchase order line. But, this is good to keep in mind when you want to apply your model in a more general context.
  We want to be able to compare the scheduled date with the expected delivery date. We want to say for example, if those dates are x days apart and criticality level of the material is y, then take action.
- **What to do with partial deliveries? Still predict one delivery date?**
  Roll it up. Look at the last goods receipt event, and then how many are being delivered so far. As you want to predict when all goods will be delivered because this is relevant for the OTIF. The partial deliveries happen quite some time.
- **What is the entity on which predictions are made?**
  The prediction should be for a scheduled line.
- **What is the prediction horizon?**
  There is not really a prediction horizon. For each open order line, there is a delivery complete indicator (open or not). You should not predict in a time frame, but just for each open order (scheduled) line.
- **How often should prediction be made?**
  Ideally daily. At the beginning of the day you want to say "for expediting, you should expedite the following order lines for me". If this is not possible due to computational time it is also possible to do it twice a week.
- **Who is the end-user? What are their objectives and ow will they benefit from the ML system?**
  The expediting team. They are looking at all orders that are overdue. You want to get to a point that there are so little orders overdue that we can act proactively in the follow up of orders. The team consists of ten FTEs.
- How are the predictions turned into value for the end-user?
  When they do not need to reschedule maintenance activities so much, and that there is a better compliance to the maintenance schedule.
- **How will we measure the success/value creation/impact of ML system (model) in production?**
  Improvement on OTIF.

Data

- **What are the data sources?**
  SAP S/4HANA and SAP ECC.
- **For how long has delivery performance been measured?**
  5 years, but there is for 3 years in Celonis.

- **How often is data updated in the system?**
  There are daily updates.

- **What is the data accuracy, and what influences this?**
  It is believed the data is accurate.

- **Are there other limitations with respect to data?**
  There is nothing that comes to mind. The dataset per material per storage location might get pretty small, however, if that is the case then we can look per supplier.

- **When I will go into the analysis of the data... which filters do I need to use, as in which data to include?**
  Exclude the internal orders from the platinum client. Moreover, only look at the purchasing of goods. Lastly, look at all data ranges (2019-now), but preferably put more weight on the recent data.