AutoML-Supported Lead Time Prediction

Enabling Smart Job Scheduling in Make-To-Order Production

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für
Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Janek Bender, M.Sc.
geb. in Osnabrück

Tag der mündlichen Prüfung:	28.10.2024
Hauptreferent:	Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova
Korreferent:	Univ.-Prof. Dr.-Ing. Detlef Gerhard

Zusammenfassung

Die verarbeitende Industrie steht immer komplexeren Kundenanforderungen in Bezug auf Produktcustomisierbarkeiten und Lieferzeiten gegenüber, während sie gleichzeitig mit Marktunsicherheiten und Störungen innerhalb ihrer Lieferantennetzwerke konfrontiert ist. Insbesondere kleine und mittlere Unternehmen aus dem Bereich der Einzelfertigung suchen nach neuen Ansätzen, um angesichts dieser Herausforderungen ihre Termintreue aufrechtzuerhalten. Durch die Fortschritte in der künstlichen Intelligenz und insbesondere im maschinellen Lernen haben Methoden zur intelligenten Planung von Fertigungsaufträgen an Bedeutung gewonnen. Die Grundlage für jeden Ansatz zur intelligenten Auftragsplanung ist jedoch eine Vorhersage der produkt-prozess-ressourcenspezifischen Durchlaufzeit, da diese eine entscheidende Eingangsgröße für den Planungsalgorithmus bildet. Dieser oft manuellen Vorhersage fehlt heute die notwendige Genauigkeit, insbesondere unter Berücksichtigung der hochgradig kundenspezifischen Produkte der Einzelfertigung.

Daher stellt diese Arbeit die erste durchgängige Methodisierung einer produkt-prozess-ressourcenspezifischen Durchlaufzeitvorhersage für kleine und mittlere Unternehmen aus der Einzelfertigung unter Verwendung von künstlicher Intelligenz in Form von automatisiertem maschinellem Online-Lernen vor. Die Methode definiert zwölf notwendige Schritte, welche den gesamten Lebenszyklus des maschinellen Lernens von der Datenaufbereitung über die Modellentwicklung bis hin zur Modellbereitstellung und -pflege umfassen. Als Kerntechnologien nutzt sie das automatisierte maschinelle Lernen, um den Ansatz für Domänenexperten zugänglicher zu machen, sowie das Online-Lernen, um Modelle zu entwickeln, die gegenüber leistungsmindernden Effekten wie Concept Drift robust sind. Die Validierung erfolgt anhand zweier Fallstudien, bei denen die mit der Methode erstellten Modelle eine Verbesserung zwischen 35% und 50% gegenüber manuellen Vorhersagen erzielten. Die bewerteten Modelle zeigten darüber hinaus die Fähigkeit, sich an künstlich in den Datenstrom eingebrachten Concept Drift anzupassen.

Abstract

Manufacturing industries face increasingly complex customer demands in terms of product customisation and delivery times while simultaneously being met with market uncertainty and disruptions within supply networks. Especially small and medium enterprises from the make-to-order domain seek new approaches to uphold their adherence to schedule in the light of these challenges. As such, methods for the smart scheduling of manufacturing jobs gained traction in recent years due to advances in artificial intelligence, and particularly in machine learning. The basis for any smart job scheduling approach however is a prediction of the product-process-resource-specific lead time as this forms a crucial input variable for the job scheduling algorithm. Today, this often manual prediction lacks the necessary accuracy, especially when considering the highly customised products of the make-to-order domain.

Therefore, this thesis provides the first end-to-end methodisation of a product-process-resource-specific lead time prediction for small and medium enterprises from the make-to-order domain using artificial intelligence in the form of automated online machine learning. The method defines twelve necessary steps spanning the entire machine learning lifecycle from data preparation via model development to model deployment and maintenance. As core technologies it leverages automated machine learning as a means to render machine learning more accessible to domain specialists and online machine learning in order to develop models robust toward performance degrading effects such as concept drift. Validation is conducted along two real-world case studies in which the models produced by the method achieved between 35% and 50% improvement over manual predictions. The models under evaluation furthermore showed the ability to cope with concept drift by adapting to adversarial changes artificially introduced to the underlying data stream.

Table of Contents

List of Figures

List of Tables

Abbreviations

ADWIN	Adaptive Windowing
AI	Artificial Intelligence
Alto	Algorithm-based Optimisation of Timely Job Control in Make-To-Order Production
ANN	Artificial Neural Network
API	Application Programming Interface
ARF	Adaptive Random Forest
ARFR	Adaptive Random Forest Regressor
AutoML	Automated Machine Learning
BN	Bayesian Network
BO	Bayesian Optimisation
CAD	Computer-aided Design
CAM	Computer-aided Manufacturing
CASH	Combined Algorithm Selection and Hyperparameter Optimisation
CBR	Case-based Reasoner
CNC	Computerised Numerical Control
CNN	Convolutional Neural Network

CSV	Comma- / Character-separated Values
(E)DDM	(Early) Drift Detection Method
DES	Discrete Event Simulation
DNN	Deep Neural Network
DT	Decision Tree
EDA	Exploratory Data Analysis
ERP	Enterprise Resource Planning
ES	Exponential Smoothing
ETO	Engineer-To-Order
FIFO	First In First Out
HPM	Human Performance Modelling
HPO	Hyperparameter Optimisation
(I)IoS	(Industrial) Internet of Services
(I)IoT	(Industrial) Internet of Things
IQR	Interquartile Range
JSS	Job Shop Scheduling
KMS	Knowledge Management System
kNN	k Nearest Neighbour
KSWIN	Kolmogorov-Smirnov Windowing
LasR	Lasso Regression
LLM	Large Language Model
LR	Linear Regression

LT	Lead Time
LTP	Lead Time Prediction
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAR	Multivariate Adaptive Regression
MES	Manufacturing Execution System
ML	Machine Learning
MLR	Multinomial Logistic Regression
MSE	Mean Squared Error
MTO	Make-To-Order
MO	Main Objective
NaN	Not-a-Number
NAS	Neural Architecture Search
NBC	Naïve Bayes Classifier
NLP	Natural Languange Processing
OPP	Order Penetration Point
OXTR	Online Extra Trees Regressor
PDM	Product Data Management
PT	Processing Time
PTP	Processing Time Prediction
PPC	Production Planning and Control

PPR	Product-Process-Resource
REST	Representational State Transfer
RF	Random Forest
RMM	Rolling Mean Model
RMSE	Root Mean Squared Error
RQ	Research Question
RR	Ridge Regression
RT	Regression Tree
SME	Small and Medium Enterprises
SNBC	Selective Naïve Bayes Classifier
SVM	Support Vector Machine
SVR	Support Vector Regression
TLT	Total Lead Time
TO	Technical Objective
TT	Transition Time
TTP	Transition Time Prediction
WAUR	Weighted Attribute Usage Ratio

1 Introduction

In Need of a Better Planning Basis for Job Scheduling

Manufacturing industries face increasingly complex customer demands in terms of product customisation and delivery times. Simultaneously, batch sizes decline and the production needs to adapt to uncertainty, flexible demands, and disruptions within the supply networks. Aside from the implications for production planners, these circumstances also pose significant challenges for shop floor operators, procurement managers, product developers, and many other positions within the value creation network.

Science, industry, and the public attempt to meet these challenges by working towards a large-scale transformation of the manufacturing industries, in Germany referred to as *Industrie 4.0* (*Industry 4.0*) [Pla22]. Other economic heavyweights put similar strategies into action such as the *Industry IoT Consortium* (until 2021 *Industrial Internet Consortium*) in the United States of America [Obj22], *Made in China 2025* [Shi17], or *Society 5.0* in Japan [DHM+20].

At its core, Industry 4.0 describes a fourth industrial revolution through the intelligent networking of machines and processes using information and communication technology. This is meant to make production more data-driven, flexible, and customer-oriented, optimise logistics, and enable a resource-efficient circular economy [Pla22]. This revolution characterises a transformation away from the (physical) product as the sole conduit for value creation and towards solutions and services, realised within globally connected value networks and monetised through data-driven business models.

In the 2030 vision for Industry 4.0 [Pla19], the authors outline three interlinked strategic fields of action for a successful realisation:

- *Autonomy* of stakeholders and market participants as a means to guarantee competitiveness in digital business models. This area covers technology development, (IT-)security, and digital infrastructure.
- *Interoperability* in terms of cooperation and open digital ecosystems as a way to bolster plurality and flexibility among participants. The term subsumes regulatory frameworks, standards, and integrational aspects as well as decentralised systems and *artificial intelligence* (AI).
- *Sustainability* as in ensuring a high standard of living for future generations through modern industrial value creation. Work and education, climate change mitigation and circular economy, and social participation make up this field.

In order to contextualise this diverse range of Industry-4.0-related activities, the *Reference Architecture Model Industry 4.0* [Pla18] as shown in Figure 1.1 was developed. It provides an architectural orientation for the most important aspects involved.

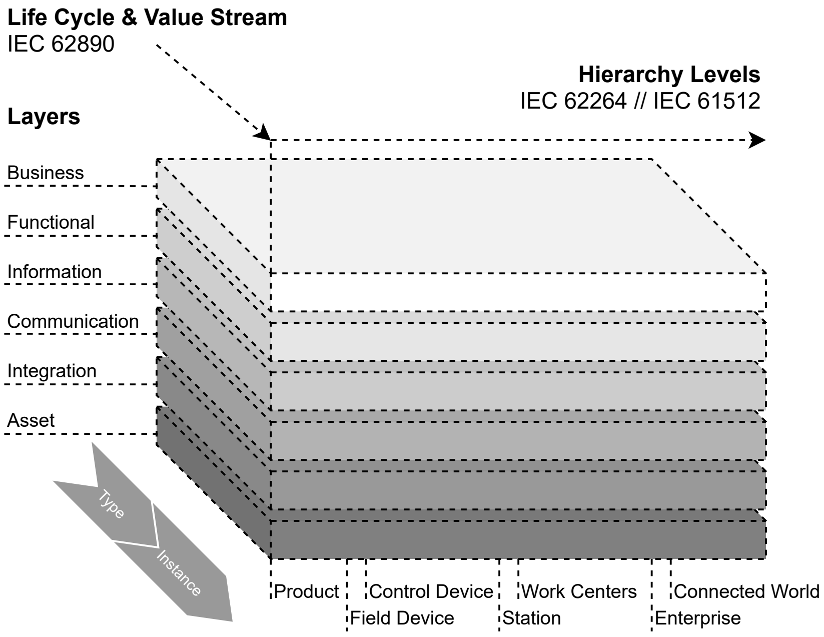

Figure 1.1: Reference Architecture Model Industry 4.0. by [Pla18, p. 8].

The first dimension on the x-axis symbolises the individual product life cycle all the way from the development or construction, maintenance and changes, through production and use, to finally, the decommissioning. The y-axis as second dimension concerns itself with the business side of things. It considers the organisation and business processes the asset is embedded in, its functions, the necessary data and access to it, the network integration as well as the physical part of the asset itself.

The third and final dimension on the z-axis describes the transformation to a new form of factory hierarchy in which systems and machines are highly flexible, with functions distributed through the network, which itself may cross individual company boundaries. This enables participants to communicate effectively with each other and directly interact across previously strictly hierarchical levels. Furthermore, the products are considered an integral part of the network and are thus not disjunct from the factory which produces them [Pla18].

Within this complex research framework of Industry 4.0, one core concept sticks out as the conduit for the shared efforts to achieve this grand vision, the *smart factory*. It is defined by acatech, the German National Academy of Science and Engineering, as:

> »A single or a consortium of corporation(s), which employ(s) information and communication technology for product development, engineering of production systems, production, logistics, and to flexibly interface with customers. The smart factory masters complexity, is less prone to errors, and increases production efficiency. Within the smart factory, humans, machines, and resources naturally communicate with each other as if they were all participants in a social network.«
>
> Translated from Promotorengruppe Kommunikation der Forschungsunion Wirtschaft - Wissenschaft and acatech - Deutsche Akademie der Technikwissenschaften e.V., 2013, [Pa13, p. 87]

Figure 1.2 contextualises the smart factory within the technological environment. With *cyber physical systems* as a basis, the globally interconnected smart factory operates within the (*industrial*) *internet of things and services* (IIoT / IIoS). Industry 4.0, and by extension the concept of the smart factory, is interdisciplinary by nature. It is thus reliant on adjacent areas of interest, such as the illustrated *smart mobility*, *smart logistics*, *smart buildings*, *smart products*, and *smart grids* [Pa13].

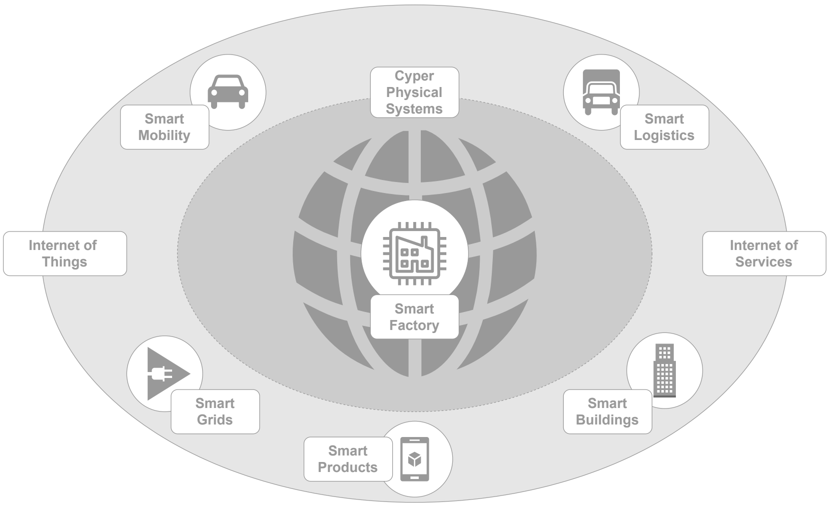

Figure 1.2: The smart factory as part of the internet of things and services by [Pa13, p. 23].

1.1 Motivation

Realisation of smart factories remains the subject of ongoing research and engineering efforts. (Partially) smart factories in the real world are the exception rather than the norm as the underlying technological and socio-economical challenges are manifold. One such challenge is in the area of intelligent job scheduling within the smart factory and its surrounding production network. This is highly relevant, especially for *small and medium enterprises* (SME), which form the backbone of the German manufacturing industries. In this context, SME are characterised by their diverse IT landscapes supporting varying degrees of automation. In addition, often producing highly customisable *make-to-order* (MTO) products in small lot sizes with tight due dates adds to the overall challenge of planning and scheduling these jobs in an optimal fashion. The production schedule is stressed further by high priority orders coming in on short notice as well as disruptions occurring on the shop floor or within the supply chain.

Contrary to the assembly line production of mass manufacturing, shop floors in this MTO environment are typically set up as a collection of individual work stations with an undirected flow of material between them. This type of shop floor architecture is known as a *job shop* [Eve02, p. 168]. Figure 1.3 displays a typical SME shop floor with various, at times redundant, work stations, loosely grouped together within the limitations of the surrounding building. The degree of automation of these work stations ranges from entirely manual (e.g. conventional turning, assembly, finishing) to highly automated (e.g. CNC-machines).

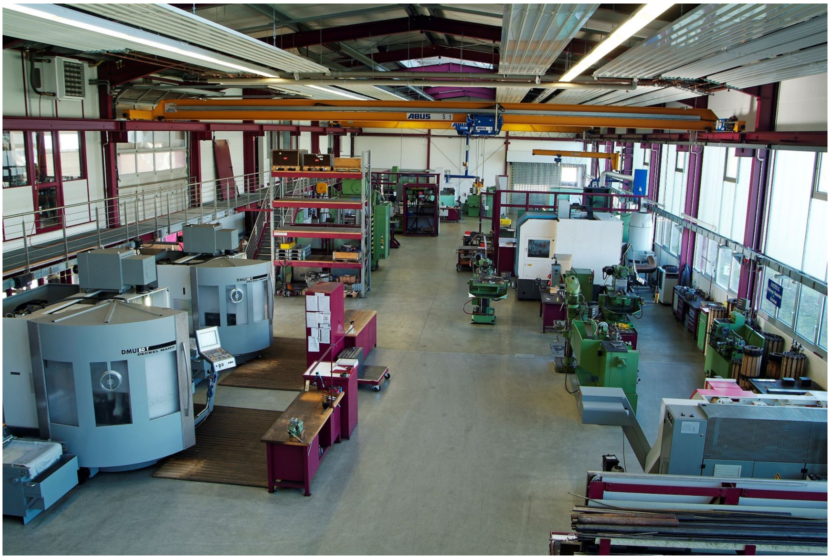

Figure 1.3: Shop floor at Breisacher Werkzeug- und Formenbau GmbH in Bahlingen a.K., Germany. Photographed within the research project Alto [GBE+22]. Photo credit Werner Breisacher.

The challenge of scheduling operations within this highly dynamic and flexible environment forms a well-known combinatorial optimisation problem, accordingly named the *job shop scheduling* (JSS) problem. An early comprehensive definition of the JSS problem can be found in [Mel66]. In summary, it describes the combinatorial challenge of scheduling O operations of J jobs on M machines in order to achieve an optimal schedule. Certain derivates of the JSS problem have been described over the years, such as the more realistic *flexible job shop scheduling* problem [ML93]. As opposed to the original, there can be alternative machines providing the same operations and multiple identical machines can be grouped together.

The optimisation goal of the JSS problem and its derivates is usually the minimisation of overall product makespan. In practice however, the optimisation goal is often multi-criterial. In addition to minimising the makespan, at times conflicting goals are the maximisation of adherence to schedule, as in minimising the violation of delivery deadlines, or in recent years, exploiting energy flexibility potentials in order to maximise the usage of renewable energy in production [FSB+23].

As observed at partnering companies within the research project Alto, SME from the MTO sector often prioritise adherence to schedule over other optimisation goals as their reputation relies on being flexible and punctual while contractual penalties for delays can be steep [GBE+22]. Add to that the aforementioned circumstances of operating in an environment where priority orders with tight due dates concerning highly customised products are arriving on short notice, optimising production for adherence to schedule becomes increasingly difficult.

Traditional and widespread rule-based optimisation approaches, such as *first in first out* (FIFO), *shortest processing time*, or the *slack time rule* [Sch06, p. 51], are not sufficient to address these complexities in today's production networks. At the same time, the recent renaissance of AI-based methods, especially from the sub-domain of *machine learning* (ML), opened new avenues to pursue novel optimisation approaches in order to lay out the technological groundwork for the smart factory. Thus, this thesis positions itself at the intersection between the problem domain of job scheduling within the smart factory and the avenue of new optimisation techniques enabled by AI, as Figure 1.4 illustrates.

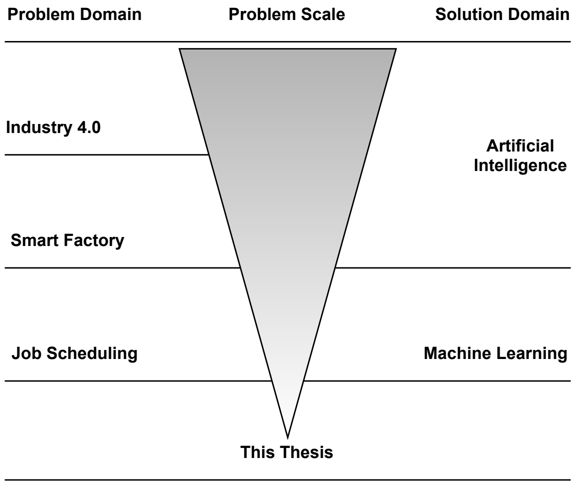

Figure 1.4: Research focus of this thesis.

1.2 Problem Statement

At the basis of any optimisation, regardless of the technique used, are the input parameters. When it comes to the optimisation of production schedules, one crucial variable is the estimated *lead time* (LT). It denotes the time a job or operation takes until it is completed. When optimising for adherence to schedule, production planners need to know the LT of a job so to not schedule it too early for it to block more urgent jobs or cause storage cost, and not too late so it will be finished before its due date. The *total lead time* (TLT) of a job as understood in this thesis is decomposed into more detail as shown in Figure 1.5.

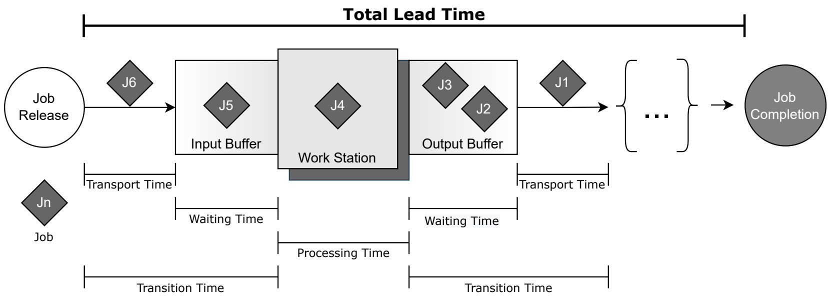

Figure 1.5: Decomposition of total lead time as understood in this thesis.

While the TLT covers the entire makespan of a job from release to completion within on a given shop floor, it is divisible further into *processing time* (PT) and *transition time* (TT). PT denotes the value-adding time a job is actually being worked on at a work station, i.e. a product being machined on a CNC-machine. TT sub-divides into *transport time* and *waiting time*. Transport time covers the time needed to move the product between work stations, i.e. by hand, through a crane, driverless transport system, or similar. Waiting time denotes the time the product spends at a work station or storage not being processed, i.e. waiting in an input or output buffer for processing or transportation respectively. Equation 1.1 formalises this straightforward relationship:

$$TLT_{job} = \sum_{i=1}^{N} PT_{operation_i} + TT_{operation_i} \tag{1.1}$$

Thus, the TLT of a job is simply the sum total of PT and TT for each operation associated with this job. For simplicity, and unless otherwise stated, LT will serve as an umbrella term for the above explained for the remainder of this thesis.

In mass production or with larger lot sizes, fairly accurate LT can be derived by measuring averages, for example as part of a *REFA time study* [REF16, p. 187 ff.]. This however is usually not applicable in MTO as lot sizes are small, products are highly individual, and operations differ from job to job.

An accurate *lead time prediction* (LTP) in an MTO scenario thus has to occur *product-process-resource-* (PPR)-specific. Where the product is self-explanatory, the resource is a machine, machine group, or indeed a human operator, and a process is what is applied by the resource to the product in order to refine it. All three elements of this PPR-model influencing LTP are visualised in Figure 1.6.

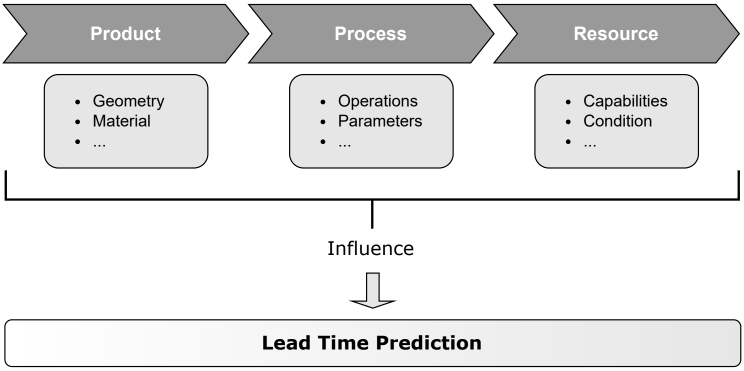

Figure 1.6: Simplified illustration of PPR-specific properties influencing LTP.

In practice, LT are estimated by production planners based on experience rather than data. Roughly 80% of companies are relying on static master data as default values [SHH+18]. These inaccurate, and in the worst case plain wrong, estimates lead to sub-optimal or even impossible schedules requiring manual actions, e.g. ad-hoc re-planning during production. This, in turn, negatively impacts previously mentioned key performance indicators such as adherence to schedule or total output. Such, regardless of the method chosen to solve the underlying scheduling problem, any optimisation algorithm relying on inaccurate input data is set up to achieve subpar performance. Hence, there is a need to improve the quality of LT predictions in order to also improve the downstream planning and scheduling tasks.

The fact that LTP in the MTO domain is so reliant on the circumstances under which it is performed also poses a chance. A data-driven approach could exploit product, process, and resource data in order to draw more accurate predictions. Fortunately, with the advent of more sophisticated data mining and AI-powered methods as well as the further digitalisation in the manufacturing sector in recent years, collecting and exploiting this data has become feasible.

1.2.1 Main Objective

Considering these three key elements, the need for more accurate *product-process-resource-specific lead time prediction* on one side, and the *availability of manufacturing data* as well as *AI-powered methods* on the other side, the *main objective* (MO) of this thesis is formulated as:

> MO: End-to-end methodisation of a *product-process-resource-* (PPR-) *specific lead time prediction* (LTP) for *small and medium enterprises* (SME) from the *make-to-order* (MTO) domain using *artificial intelligence* (AI).

Figure 1.7 illustrates the MO laid out above. Information about the product, its associated production processes, and resources on the shop floor capable to implement these processes is to be exploited in order to achieve a more accurate LTP, which in turn serves as an enabler for smart scheduling approaches.

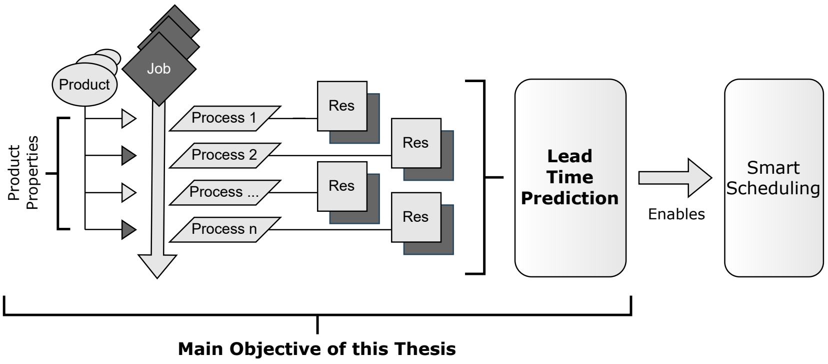

Figure 1.7: PPR-specific LTP as an enabler for smart scheduling.

1.2.2 Research Questions

Further systemising the MO, the following *research questions* (RQ) serve as a guideline for this thesis:

Data-related
RQ1.1: What data regarding products, processes, and resources available in SME is suitable for LTP?
RQ1.2: How does this data need to be (pre-)processed in order to serve as input for an AI-system?

AI-related
RQ2.1: Which AI-methods are suitable to generate predictive models realising LTP?
RQ2.2: How can these predictive models be protected from performance degradation over time?

Integration-related
RQ3.1: How can the deployment of these predictive models be integrated into an SME's system landscape on a technical level?
RQ3.2: How can the application of this AI-method be integrated into an SME's production planning and control on an organisational level?

1.3 Structure of this Thesis

In order to achieve the MO and contribute to answering the research questions, this thesis is structured as pictured in Figure 1.8. After the previous introduction and problem description as well the formulation of the objectives and research questions in Chapter 1, the managerial and technical foundations are laid out in the background in Chapter 2. This covers basics in production as well as AI, especially automated machine learning.

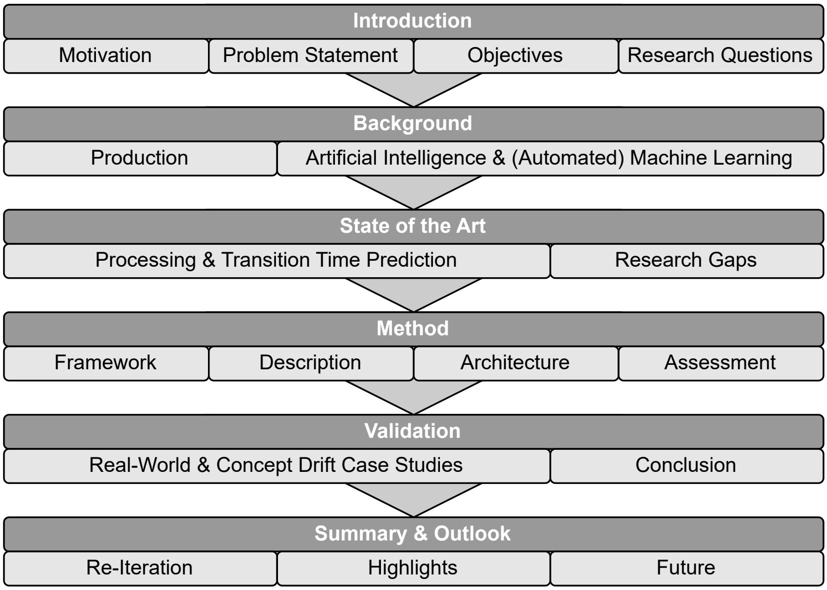

Figure 1.8: Structure of this thesis.

This is succeeded by a more narrowly focused, systemised view on the state of the art in Chapter 3, showcasing relevant works on processing and transition time prediction. Research gaps are highlighted as well. Describing the core contribution of this thesis, Chapter 4 introduces a method for AI-supported LTP. After laying out the framework in which the method is operated, it is described, and a system architecture is sketched out. The method is theoretically assessed and differentiated from the state of the art. In Chapter 5, it is validated against two real-world case studies. Case study A provides unaltered real-world data while in case study B, an artificial concept drift is introduced. The results of both case studies are used to draw a conclusion on the practical ability of the method to fulfil the main objective of this thesis. Finally, Chapter 6 re-iterates the work conducted in thesis, highlights the results and contributions, and provides an outlook into the future.

2 Background

The Managerial & Technical Foundations

Following the main objective and the associated research questions laid out in Section 1.2, important fundamentals have to be established. The first part of this chapter introduces the foundations of inner-company production planning and control practices under which the main objective has to be achieved. In contrast, the remainder of this chapter outlines artificial intelligence as understood in this thesis with a special focus on automated machine learning.

2.1 Production

Production means the creation of goods through the combination of the factors of production: land, labour, and capital. It denotes a transformative process in which the inputs, the factors of production, are used to generate an output, the goods. Thus, it contributes directly to the value-adding activities of a company [KLL22, p. 3]. In this thesis, the terms production and manufacturing are used interchangeably.

2.1.1 Production Types

Production can be divided into different types along varying criteria. Two type schemes relevant to this thesis are laid out here. Schuh and Schmidt define a type scheme based on lot size and repetition rate, i.e. how many products are produced per job and how many jobs concerning the same product occur within a given time period [Sch06, p. 129-130].

Their scheme covers four types of production laid out in Table 2.1 with the make-to-order (MTO) focus of this thesis highlighted.

Table 2.1: Types of production according to [Sch06, p. 129-130].

Type	Lot Size	Repetition Rate
One-Time	*small*	0
Make-To-Order & Small Series	**< 50**	**< 12**
Series	> 50	< 24
Mass	*very large*	*continuous*

Kellner et al. describe a differentiation based on the position of the *order penetration point* (OPP) [KLL22, p. 9]. According to Olhager, the OPP marks the point in the value chain where a given product is linked to a specific customer. It is therefore also referred to as the customer decoupling point [Olh03]. Figure 2.1 illustrates typical positions of the OPP in companies' value chains, highlighting the different types of production.

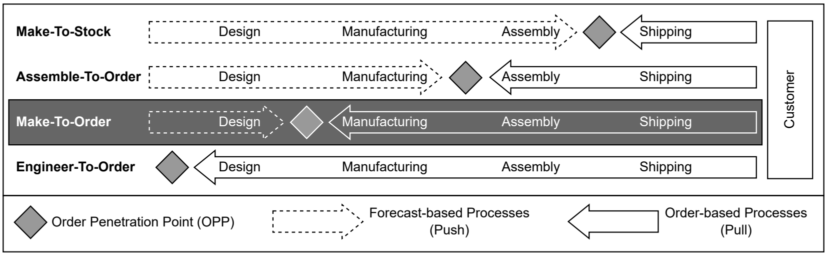

Figure 2.1: OPP-based types of production according to [KLL22, p. 9] based on [Olh03]. The MTO focus of this thesis is highlighted.

On one end, a late OPP favours standardised products made in bulk according to market forecasts and with limited customer input. On the other end, (one-time-)products are engineered and produced directly to specification, involving the customer into the value chain early on. This thesis is aimed at the MTO type of production where lot sizes and repetition rates are modest and the OPP is located in the value chain prior to the manufacturing stage.

2.1.2 Production Organisation

The production can be organised in various ways, depending on the requirements and constraints set by, among others, the production type. This organisation relates to the spatial positioning of resources as well as material flow between those [Sch06, p. 130-131]. Schuh defines four distinct production organisations. Extended by another one by Greschke et al., five organisations are exemplary shown in Figure 2.2.

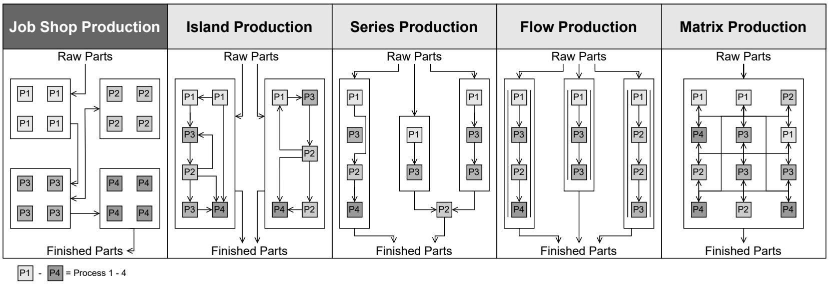

Figure 2.2: Different production organisation models based on [Sch06, p. 131] and [GSTH14]. The job shop focus of this thesis is highlighted.

The *job shop production* is characterised by similar resources being grouped together, for example as machine groups. The material flow is undirected and already visited resources can be revisited. Contrary to this, in the *island production*, resources are grouped together in a product-oriented fashion. The material flow is again undirected though individual islands largely act autonomously with great control over it. In the *series production*, resources are set up based on semi-finished part groups. The material flow is directed, however individual resources may be skipped as needed.

Compared to the prior, the *flow production* enforces a rigid directional material flow which is usually clocked [Sch06, p. 130-131]. Finally, the *matrix production* as a more recent concept is aimed at realising high product variance in simultaneously high volumes. It does so by being highly flexible on an unclocked schedule. However, this approach also raises complexities in planning and resource allocation [GSTH14].

As previously mentioned, the choice of production organisation is often dictated by the company circumstances and the production type. According to Kellner et al., it can be generalised that a production with low complexity and variability but high volume and degree of standardisation benefits from a flow production. Contrary to that, a production with high complexity and variability but low volume and degree of standardisation is more suited to a job shop production [KLL22, p. 84]. The latter sets the frame of this thesis with the contemplated companies operating in an MTO fashion typically organise their production in a job shop.

2.1.3 Production Planning & Control

In order to facilitate the production, methods of *production planning and control* (PPC) are employed. Production planning defines the subject and processes of the production. Contrary to this, production control governs the execution of these processes as part of the order fulfilment. According to Schuh and Roesgen [Sch06, p. 28] as well as Kellner et al. [KLL22, p. 160], PPC strives to fulfil the following goals:

- High adherence to schedule and readiness to deliver,
- high flexibility and smooth capacity utilisation,
- short lead times,
- low capital tie-up,
- low stock and procurement cost.

In the short term, it attempts to do so by generating production schedules aiming at fulfilling these, at times conflicting, goals as best as possible [KLL22, p. 161]. In the MTO scenario, a production schedule can be viewed as a fluent set of jobs, each with its individual routing file. In conjunction with the bill of materials, this defines the timed allocation of resources (e.g. raw materials, machines, labour) and the operations or production processes to execute in order to fulfil the job [KLL22, p. 156-158]. In the remainder of this thesis, the terms operation and process are used interchangeably. Table 2.2 shows a simplified exemplary routing file featuring five operations.

Table 2.2: Simplified exemplary routing file based on [KLL22, p. 158].

Some Exemplary Part	Job ID: 2023-42-A	Routing File ID: 58390
Due Date: 01.09.2023	Drawing ID: 135-246	Material: 1.4104
Raw Weight: 6.2kg	Fin. Weight: 4.0kg	Quantity: 1
Operation	**Machine Group**	**Est. Processing Time**
01 Sawing	MG-47-A	35 min.
02 Forging	MG-52-E	126 min.
03 Rolling	MG-25-B	14 min.
04 Hardness Testing	MG-77-F	5 min.
05 Packaging	MG-11-C	10 min.

IT-systems to support PPC and especially the generation of production schedules are often using the *enterprise resource planning* (ERP) as a basis and reside under various terms according to their respective foci. This includes PPC-systems, *manufacturing execution systems* (MES), and *advanced planning and scheduling systems* [KLL22, p. 373-375]. However, in SME, which are the primary focus of this thesis, system landscapes are heterogeneous and especially in the MTO domain, PPC is often carried out manually with only limited support from the above-mentioned systems.

2.2 Artificial Intelligence

The term *artificial intelligence* (AI) was first coined in the 1950s [MMRS55]. A multitude of different definitions of AI have emerged since. Legg and Hutter provide a comprehensive overview of definitions from researchers of various fields [LH07]. With regards to this thesis, three exemplary definitions by AI researchers are highlighted below. Starting with Nilsson who simply states that:

> »AI [...] is concerned with intelligent behaviour in artefacts.«
>
> Nils J. Nilsson, 1998, [Nil98]

Whereas Poole et al. more concretely define AI as:

> »Computational [(artificial)] intelligence is the study of the design of intelligent agents. An intelligent agent is a system that acts intelligently: What it does is appropriate for its circumstances and its goal, it is flexible to changing environments and changing goals, it learns from experience, and it makes appropriate choices given perceptual limitations and finite computation.«
>
> David L. Poole et al., 1998, [PMG98, p. 1]

In their standard textbook on AI, Russel and Norvig discuss what (artificial) intelligence entails at great length and conclude that:

> »[...] intelligence is concerned mainly with rational action. Ideally, an intelligent agent takes the best possible action in a situation. [Artificial intelligence studies] the problem of building agents that are intelligent in this sense.«
>
> Stuart J. Russell & Peter Norvig, 2016, [RN16, p. 30]

In order to present a more concrete idea of AI, Figure 2.3 illustrates the many application domains AI covers such as but not limited to *game playing*, *automated reasoning and theorem proving*, *expert systems*, *natural language processing* (NLP), *human performance modelling* (HPM), *planning and robotics*, as well as *machine learning* (ML) [Lug09, p. 20 ff.].

In relation to this thesis, ML is the most relevant application domain of AI hence it is highlighted and discussed in detail in the subsequent section.

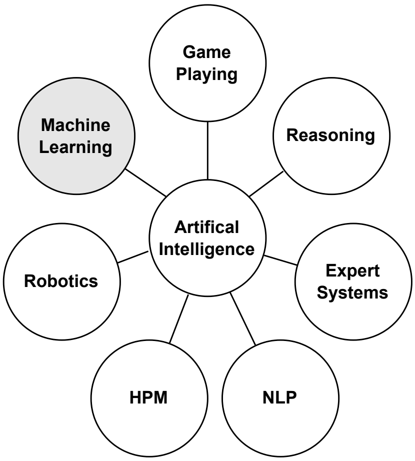

Figure 2.3: Application domains of artificial intelligence according to [Lug09, p. 20 ff.].

2.2.1 Machine Learning

Similar to the history of AI, the origins of ML lie within the 1950s where Samuel first investigated how computers could be set up to solve the game of checkers without being explicitly programmed to by employing the *process of learning* [Sam59]. A widely accepted formalised definition of ML was decades later coined by Mitchell:

> »A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.«
>
> Tom Mitchell, 1997, [Mit97, p. 2]

ML usually addresses tasks to which solutions are difficult or unpractical to programme manually. Instead, experiences in the form of data are used to learn from and develop a model which is able to solve the task. This is done using a learning algorithm and this process is referred to as *training*. In order to assess the ability of the model to solve the problem, task-specific performance measures are computed and tracked accordingly. Popular performance measures are often error-distance-based, i.e. they measure the difference between a model's predictions and the *ground truth* [GBC16, p. 99-105].

The challenge in ML is for models to maintain a constant performance on new, previously unseen data, which is referred to as *generalisation*. During training, the performance measure is used to determine and minimise the training error. Once the model is confronted with data is has not seen during training, the testing error can be computed. The resulting difference between the two error measures provides an idea on how well the model generalises. The ability of the ML setup to minimise both the training error and the difference between the training and testing error determines how well it will perform on the task in practice. The circumstance of an ML algorithm failing to achieve a sufficiently low training error is called *underfitting*. On the other hand, *overfitting* occurs when an ML algorithm suffers from a large gap between training and testing errors [GBC16, p. 110-113]. Figure 2.4 visualises the described.

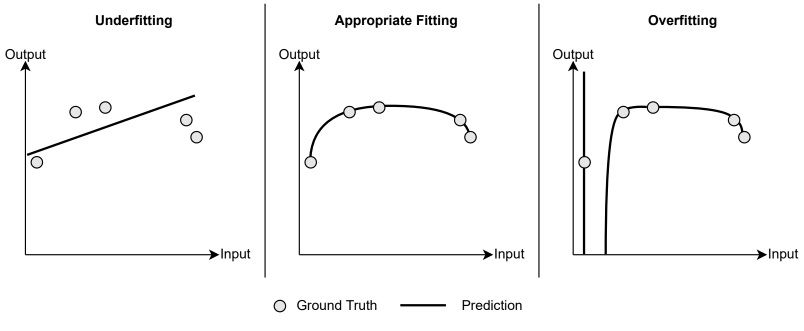

Figure 2.4: Underfitting, appropriate fitting, and overfitting. Based on [GBC16, p. 113].

The input axis corresponds to the predictors, i.e. the variables to draw a prediction from, while the output axis refers to the target variable to be predicted. Inputs or predictors are also called features. The aforementioned ground truth is also referred to as label in the context of the training. In the remainder of this thesis, the respective terms are used interchangeably.

Types of Learning

ML is subdivided into different types of learning. These types can be broken down further into *learning problems* and *techniques*. There are three distinct learning problems relevant to this thesis:

- *Supervised learning* describes the approach of using a labelled historical dataset, which contains both the predictor variables as well as the target variable, to train and test a predictive model [GBC16, p. 105]. Each data point is represented as a *feature vector*, a numerical representation of the data [ZC18, p. 3]. It is of fixed size and is tied to a corresponding target variable or label [Bis06, p. 3]. The term supervised stems from the idea that there is some *teacher* (e.g. a human expert) who provides the ground truth (label) to the algorithm so it can learn the correct relations from it. Before training, the data is separated into training and test sets. During training, an algorithm will be fed with both the training set's feature vectors and their corresponding labels. In the subsequent stage, the trained model will be presented with the previously unseen feature vectors from the test set on which it will make predictions. These predictions are then compared against the test set's labels in order to measure the model's ability to generalise [RN16, p. 696]. In practice, obtaining labelled data can be a challenging task, often only achieved through manual labelling by human experts. A supervised learning problem comes in two flavours. A *classification* is present when the labels fall into discrete categories, e.g. when categorising car shapes into sedan, convertible, estate, etc. A special case is the binary classification where there are only two classes, 1 or 0. For example, when trying to detect whether a given image contains a cat (class 1) or not (class 0). A *regression* is present when the target variable is continuous, e.g. when predicting the oil temperature of an engine under different running conditions [RN16, p. 696 ff.].

In some cases, a regression may be discretised to a classification by binning continuous variable ranges into fixed-size categories. An example would be predicting the age bracket ([0-12], [13-18], [19-35], etc.) given individuals belong to. The problem of lead time prediction as an ML task can be modelled as either a regression or a classification which is further discussed later in Section 3.1.

- *Unsupervised learning*, as opposed to its supervised counterpart, deals with unlabelled data [GBC16, p. 105]. The most common unsupervised learning task is *clustering* in which the algorithm tries to find clusters of similar data points [RN16, p. 694-695]. As an example, a model might cluster data about car shapes instinctively into the above mentioned classes without explicitly being told to, just by grouping together similar data points. Another useful example from the production domain would be the condition monitoring of an engine, where an unsupervised learning algorithm tries to detect anomalies in the machine's parameters at any given time, highlighting potential faults [STB+22].

- *Semi-supervised learning* is a special case falling between the abovementioned. This problem type occurs when there is some part of the data which is labelled and another, usually larger, part is unlabelled [GBC16, p. 243-244]. And even the labels which are present are not to be trusted universally as inaccuracies beyond noise may occur. For real-world problems, a clear-cut distinction between supervised, semi-supervised, and unsupervised is rarely achieved [RN16, p. 695]. Especially with heterogeneous shop floor data from different sources, in varying quantities and qualities, and often collected with human involvement, semi-supervised as opposed to supervised learning has to be expected.

The literature defines more learning problems and variations besides the ones mentioned above. Detailed information on these can be found in textbooks such as [Mit97, Vap98, Bis06, HTF09, Mur12, RN16, GBC16, WFH16, SB18, MRT18]. Figure 2.3 summarises the previously discussed learning problems as relevant to this thesis with the main focus on (semi-)supervised and reinforcement learning.

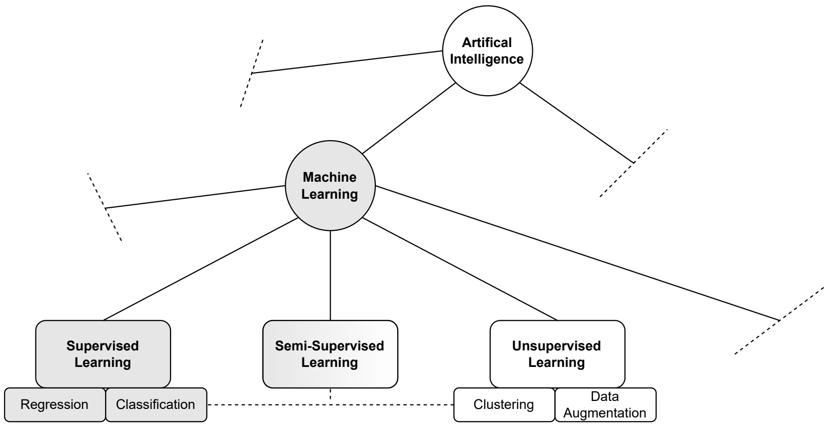

Figure 2.5: Learning-problem-oriented view of machine learning as relevant to this thesis.

Learning Techniques

Complementary to the learning-problem-oriented view on machine learning, a technique-oriented view can be established. Again, the entire palette of learning techniques can be found in the above-cited textbooks. For this thesis, four techniques stick out and are briefly laid out below:

- *Offline learning* (or *offline training*, *batch learning*, *batch training*) is the standard approach in *traditional* ML and describes the practice of training a model on a finite amount of data collected at some point in time [Mur12, p. 261]. In a supervised learning scenario in its simplest form, the data is divided into a training and test set. The training set is used to train the model using some algorithm and the test set, containing data not yet seen by the model, is then used to assess its performance on the given learning task. It is thus the simplest and most intuitive technique to perform machine learning. However, during deployment the model might encounter data that differs in its statistical properties from the data the model was originally trained on and, since the model cannot simply adjust itself to these changes, its performance degrades over time. This is known as *concept drift* [GŽB$^+$14], which is discussed in detail in Section 2.2.3.

In this offline scenario, the only way to cope with concept drift is to collect sufficient amounts of more recent data and retrain the model from scratch. Offline training might also be unsuited when there is simply too much data to fit into memory at once or when dealing with continuous data streams, especially time-series-data. In the remainder of this thesis, offline / batch learning / training are used synonymously.

- *Online learning*, (or *online training*, *incremental learning*, *incremental training*) as opposed to offline learning, governs a learning approach where the model is trained incrementally on a case-by-case-basis [HTF09, p. 397]. This means that data points are fed to the algorithm individually, often in a continuos data stream. Thus, the model can be trained even during deployment as it first predicts on a given data point and is eventually told the ground truth which triggers an (ever so slight) adjustment to the model [RN16, p. 752-753]. This approach has two key advantages. For one, it is often used when the data volume is so great, it simply would not fit into a given system's memory all at once. Hence the incremental approach allows the data to be fed as a stream and thus never exceeds the system's memory capacity. Another advantage lies in the fact that the model is able to continuously learn and improve its predictive power, and is thus, within limits, capable of adapting to shifts in the underlying data [MRT18, 177-206]. Especially the second advantage of *lifelong learning* makes online learning a very interesting method to investigate for the ever-changing nature of problem spaces in the context of a shop floor, especially within highly customised MTO production. In the remainder of this thesis, online learning / training and incremental learning / training are used synonymously.
- *Ensemble learning* combines a collection of multiple models to solve a problem [RN16, p. 748-752]. This approach is predicated on the idea that, under given circumstances, a set of simpler models joined together outperforms a singular, more complex model [HTF09, p. 605]. The ensemble model is achieved often by weighting the individual models' outputs and then averaging them together [GBC16, p. 472]. Other popular methods include but are not limited to *boosting* [MRT18, p. 145-170], *bagging* [RN16, p. 760-762], and *stacking* [HTF09, p. 290].

A key advantage of ensemble learning is that individual models can be trained comparatively cheaply in a highly parallelised fashion, making them suitable for periodic or event-driven retraining in order to maintain model quality.

- *Deep learning* describes a method centred around *artificial neural networks* (ANN). A *deep neural network* contains *many* hidden layers [Ben09]. Within these layers, multiple *simple* networks can be intelligently wired together in order to form a *complex* network [GBC16, p. 1-2]. These complex networks can be deployed to tackle complicated learning problems which traditional machine learning techniques would not be able to solve. DNNs frequently make headlines, for example when IBM's Deep Blue beat chess champion Gary Kasparov in 1997 [CHH02], or AlphaGo which mastered the game Go [SH16], and more recently the powerful GPT *large language models* (LLM) [BMR+20]. While deep learning may be able to address complicated learning problems, it is an equally resource-hungry endeavour usually requiring vast amounts of data, computational resources, and time [GBC16, p. 19-22].

2.2.2 Feature Engineering

The starting point for any learning technique however is the data, which is often not in a shape suited for ML [Cad17, p. 13]. In order to produce data an algorithm can successfully learn from, several important steps have to be completed in order to transform this raw data into features. This process is called *feature engineering*, often also associated with data pre-processing, and covers a range of activities [ZC18, p. 3]. A first step forms the *exploratory data analysis* (EDA) with the goal of establishing a fundamental understanding of the data [Tuk77]. This is facilitated mostly by determining statistical properties and producing human-readable visualisations. It is used to drive the understanding of the data and discover relationships among it, providing a basis for the downstream feature engineering activities. Figure 2.6 provides a few selected examples of popular visualisation techniques.

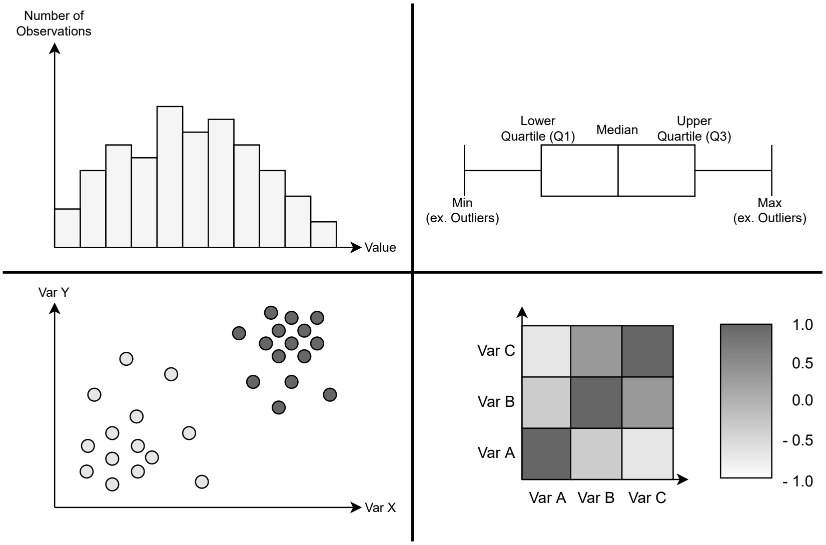

Figure 2.6: Examples of data visualisation techniques. Clockwise top left to bottom left: histogram, box plot, correlation matrix, and scatter plot. Based on [Tuk77].

Following the EDA, the actual transformation from data to features may commence. This includes cleaning the data of unwanted or erroneous data points as well as handling missing values. Missing values are most commonly addressed either by outright deletion of the affected data points [All01, p. 6-8] or imputation of values, for example by replacing missing values with means [Lit12, p. 69-72]. There are essentially two types of data: *numerical* or *continuous*, and *categorical* or *qualitative*, which require different approaches. Numerical variables often differ in magnitude, are skewed, or contain extreme values which negatively impacts learner performance [Kuh20]. Scaling data to counter magnitude issues can be achieved through *standardisation* or *normalisation*. In standardisation, from every variable value the variable's mean is subtracted and the result is divided by the variable's standard deviation [GE06]. This is shown in Equation 2.1.

$$x'_i = \frac{(x_i - \mu_i)}{\sigma_i} \tag{2.1}$$

In normalisation, each value is subtracted by the variable's minimum and the difference is divided by the variable's maximum subtracted by its minimum [AG19, p. 40]. This is shown in Equation 2.2.

$$x'_i = \frac{(x_i - x_{min,i})}{(x_{max,i} - x_{min,i})} \tag{2.2}$$

Another popular preprocessing technique on numerical data is *binning*, whereby a continuous variable is discretised as values are replaced by their assigned bins. Binning may be conducted on the basis of fixed widths or quantiles. An example would be the binning of a continuous age variable into discrete age bins (0-12, 13-17, ..., 75+ years old) [ZC18, p. 10-15]. Depending on the individual learning problem, more advanced techniques, such as Fourier transformation, extraction of local features, linear and non-linear space embedding methods, or non-linear expansions may be required to preprocess numerical data [GE06].

The second type of data, categorical one, is often represented through strings, which most learning algorithms cannot process. An example would be country names or assessments such as "good", "neutral", "bad". Transforming categorical into numerical data is achieved through *encoding* [Kuh20]. There exist numerous schemes to encode categorical data with various impacts on model performance, depending on the learning problem and chosen algorithm. Two of these, the *label encoder* and the *one-hot-encoder* [ZC18, p. 78-79], are exemplary shown in Figure 2.7.

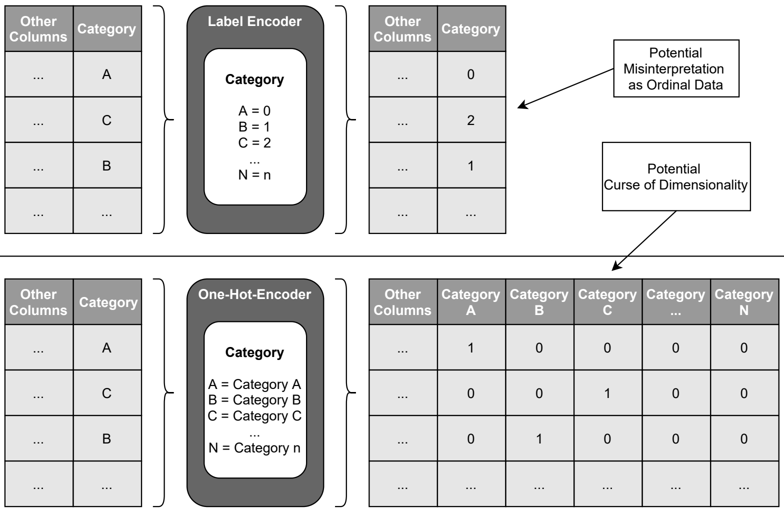

Figure 2.7: Label encoder on the top versus one-hot-encoder [ZC18, p. 78-79] at the bottom.

A label encoder is essentially a key-value-map where each unique category value for any given category column in the data is assigned a number. During the encoding stage, every categorical value encountered is transformed into its numerical counterpart. This is one of the simplest form of encoding, both quick to execute and easy to interpret. However, some ML algorithms might learn an ordinal pattern from the encoded data which is not intended. It could thus lead to false correlations being discovered by the model, thereby impacting model performance. As an alternative on the opposite end of available encoding methods, the one-hot-encoder circumvents this problem. It does so by removing the original category column while adding a new column for each unique categorical value encountered. Each of these new columns contains a strictly binary value, 0 for when the category is not present in the original feature vector, and 1 when it is [ZC18, p. 78-79]. This prevents the ordinal problem of the label encoder by providing a distinct feature for every category value but does so at the cost of increasing the overall dimensionality of the data. In turn, the increase in dimensionality might negatively affect training performance as it leads to slower convergence. Even worse, it could lead to overfitting the model. This phenomenon is referred to as the *curse of dimensionality* [Bel03].

There exist numerous approaches to mitigate this problem through dimensionality reduction achieved by *feature selection*. Three principal approaches in this regard are *filters*, *wrappers*, and *embedded methods*, as shown in Figure 2.8.

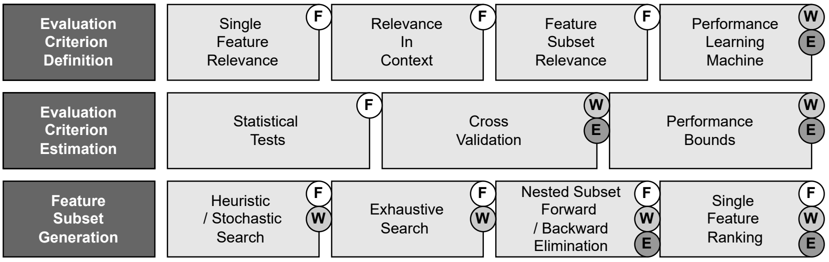
Evaluation Criterion Definition
Single Feature Relevance
F
Relevance In Context
F
Feature Subset Relevance
F
Performance Learning Machine
W
E
Evaluation Criterion Estimation
Statistical Tests
F
Cross Validation
W
E
Performance Bounds
W
E
Feature Subset Generation
Heuristic / Stochastic Search
F
W
Exhaustive Search
F
W
Nested Subset Forward / Backward Elimination
F
W
E
Single Feature Ranking
F
W
E

Figure 2.8: Filter (F) vs. Wrapper (W) vs. Embedded Methods (E). Based on [GE06]. Illustration inspired by [Sau20, p. 52].

Filter, wrapper, and embedded methods are distinguished by which criteria are used to evaluate features, how these criteria are estimated, and what search strategy for subset generation is employed. Filter methods function independently from the model, using statistical tests, such as correlation coefficients [GE06]. Correlation between features and outputs indicates strong features while correlation among features should be minimised where possible [LCW$^+$18]. Different methods include feature selection through the *Pearson correlation coefficient* [Cle11, p. 107-111], the *Spearman rank correlation* [Cle11, p. 113-119], the *Chi*^{2} *score* [LS95], the *Fisher score* [GLH11], or methods based on *mutual information* [BPZL12, LCW$^+$18].

Wrapper methods use the model performance metrics to determine a feature subset and are thus dependent on the learning algorithm. Cross validation is a popular wrapper technique where, in its simplest form, the data is divided into arbitrary subsets of which one is used for testing and the others for training. Subsets are then rotated and run until every subset has been used for testing once. The total error is computed by averaging the errors of each run [GE06].

Embedded methods are similar to wrappers but are directly integrated into the learning algorithm. All methods can utilise search strategies to optimise the process of discovering a sufficient subset of features. Furthermore, hybrid approaches are possible whereby filers and wrappers or embedded methods are combined in the learning system [GE06].

In an online learning scenario however, where data is arriving incrementally, many of the aforementioned methods do not function properly as they are tailored to offline learning where the entire dataset is available upfront [GRB+19]. A particular problem in this regard occurs when features become increasingly or decreasingly important to the learning algorithm over time, which is referred to as *feature drift* [NWNW12]. This is not to be confused with the aforementioned concept drift. In practice and where possible, running statistics can be applied instead of static ones to realise filter methods [MHM+21]. Examples for more sophisticated algorithmic approaches would be *fast online streaming feature selection* (Fast-OSFS) [WYD+13] and the *scalable and accurate on line approach for feature selection* (SAOLA) [YWDP14]. However, Gomes et al. note in a survey paper that the state of the art is not sufficient and new methods for feature selection in data streams, particularly focused on adapting to feature drift, must be developed in the future [GRB+19].

2.2.3 Concept Drift Handling

As previously mentioned in Section 2.2.1, in online learning a phenomenon called concept drift may be encountered. Gama et al. state that a concept in supervised learning describes the joint distribution $P(y|X)$ where X are the features and y the corresponding target variable. A concept drift occurs when this joint distribution changes in a way that renders a model predicting y from X unable to perform this task or it suffers significant performance loss doing so [GŽB+14]. This may creep in over time, thus it cannot be foreseen during model development. An example from the manufacturing domain could be a change in the shop floor configuration, e.g. the introduction of some more efficient production process or resource resulting in lower lead times. A fact which is unknown to any model trained on data from the old shop floor configuration and potential cause for less accurate predictions ultimately resulting in performance degradation.

Distinctions can be made with regards to the impact a concept drift might have on a predictive model. Gama et al. differentiate between two types of drift as shown in Figure 2.9. The left side of the figure illustrates the original data distribution a model distinguishing two classes was trained on. In the middle, *real concept drift* occurs as the data distribution P(y|X) is shifting, moving the decision boundary and thereby invalidating the model. The right side shows what is called a *virtual drift* in which P(X) changes but P(y|X) is unaffected. The data distribution has changed but the decision boundary has not moved and the model remains functional [GŽB+14].

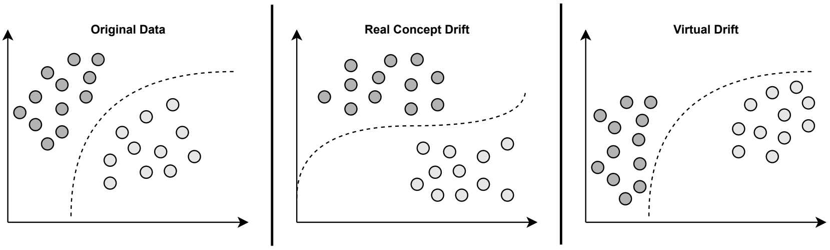

Figure 2.9: Types of drifts where circles represent instances, colours represent different classes, and the dotted line indicates the decision boundary. Original illustration by [GŽB+14].

Within the subdomain of real concept drift, distinctions can be made to the concept drift patterns as shown in Figure 2.10. The figure visualises how the mean value of a given set of data shifts over time under different concept drift patterns. Sudden concept drift occurs abruptly, showing a clear offset in the mean. Incremental concept drift creeps in over time. Gradual concept drift bounces back and forth for some time but ultimately converges on a stable new concept. Lastly, reoccurring concept drift exhibits repeating shifts over time which level back to the original mean before reoccurring. In reality of course, a distinction between these patterns may not be clear cut and patterns might evolve even further over time. As Webb et al. note, especially gradual concept drift is difficult to define and characterise in reality [WHC+16]. It is also worth pointing out that individual outliers are not considered concept drift in this sense [GŽB+14].

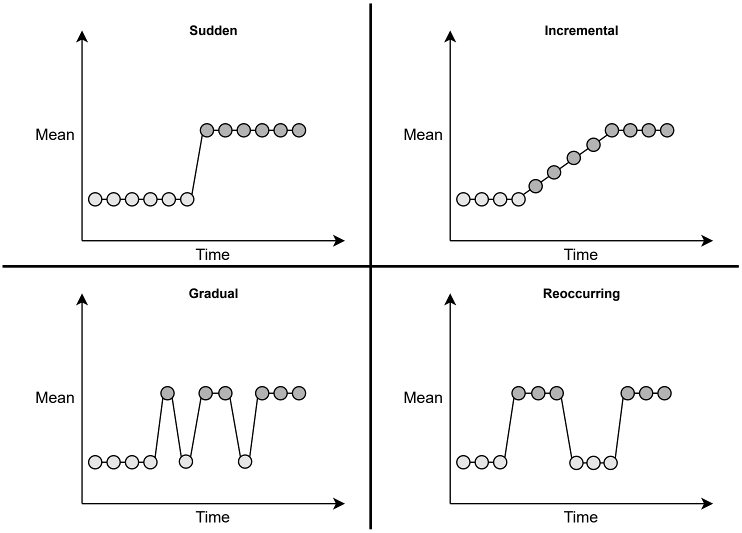

Figure 2.10: Concept drift patterns. Based on [GŽB+14].

Handling concept drift is approached via two mechanisms. The *detection* mechanism's task is to identify concept drift as soon as it occurs. The detection has to be robust enough to distinguish drift from outliers and noise. When necessary, the *adaptation* mechanism then alters the underlying model in a way so that it retains its performance on the learning problem at hand [GŽB+14].

Concept drift detectors monitor the input data stream and / or model-related performance criteria. They are typically based on methods revolving around *sequential analysis*, *statistical process control*, *monitoring distributions on two different time-windows*, or *contextual methods*, which combine different approaches [GŽB+14]. One of the earliest drift detectors is based on the *Page-Hinkley-Test* which tracks the mean value of the observed variable and signals drift detection once a given threshold is passed [Pag54]. Another more recent and popular method is the *Drift Detection Method* (DDM), which monitors the model's error rate [GMCR04]. DDM was later expanded into the *Early Drift Detection Method* (EDDM) which operates on running statistics and shows improved detection capabilities, particularly on gradual concept drift [BGDF+06].

An example for a modern drift detector popular among practitioners is the *ADaptive WINdowing* (ADWIN) algorithm [BG07]. As the name suggests, ADWIN maintains a flexibly-sized sliding window of the most recent data points. As long as the observed variable's distribution remains static, the window continues to grow in order to retain a high enough variance in the observed data. The actual detection is realised by dividing the window into two sub-windows and computing mean values of the observed variable for both. Once the difference between the mean values in these sub-windows surpasses a given threshold, i.e. concept a drift is detected, the older sub-window is discarded, slashing the window in half. As the distribution stabilises, the window continues to grow again. Using separate detector instances with different thresholds, a warning-detection-scheme can be implemented, whereby early warnings of potential concept drifts can be issued while only actually adapting a model once a significant enough concept drift occurred. As an alternative, the *Kolmogorov-Smirnov Windowing* (KSWIN) drift detector also relies on sliding windows, though fixed in size, but applies Kolmogorov-Smirnov tests on two subsequent windows in order to identify concept drift [RHS20]. The choice of detector is ultimately problem-specific though it is possible to form detection-ensembles by combining multiple detectors [TO23].

Model adaptation can be conducted under two broad strategies. *Blind* adaptations are proactive and alter a model without explicit detection of a concept drift. This is typically achieved by periodic retraining based on fixed-sized sliding windows. Note that most online learners can be considered as blind adaptation as models are adjusted to the most recent data. *Informed* strategies are reactive by nature and require a trigger, e.g. a concept drift detector signal, to be implemented. Adaptations either come in the form of a *global replacement*, whereby the existing model is discarded and retrained entirely. Or as a *local replacement*, whereby only some aspects of the existing model are altered [GŽB+14]. An example for the latter is the *Hoeffding tree algorithm*, which realises an incrementally learning decision tree [DH00].

Given the node structure of the tree, the detector can be used to identify the node at which a concept drift occurred. Based on the observation that nodes near the tree root were generated from older and nodes near the leaves from newer data, the adaptation mechanism then pushes up the newer information from the leave nodes. Pruning the leaves afterwards, it therefore *forgets* information from the time before the concept drift occurred [GŽB+14]. Which detector-adaptation scheme to use is ultimately problem-specific and not discussed further at this point.

2.2.4 Automated Machine Learning

Developing ML models is a laborious and tedious work characterised by its largely experimental nature. Repetitive tasks, such as *algorithm selection* or *hyperparameter optimisation* (HPO), are often carried out in a trial-and-error fashion. At the same time, the growing need for data-driven solutions is not matched by the supply of ML experts. In a 2015 article in the Harvard Business Review, Frankel appropriately condensed this problem down to:

> »Data Scientists Don't Scale«
>
> Stuart Frankel, 2015, [Fra15]

In this article, Frankel argues that human-powered data science is simply not a scalable solution as there are not enough experts on the labour market. Hence the automation of tasks within data science, and consequently ML, becomes a necessity. This need has also been recognised by the scientific community, leading to the subfield of *automated machine learning* (AutoML). In an early work by Feurer et al., the authors provided a comprehensive definition:

> »We define AutoML as the problem of automatically (without human input) producing test set predictions for a new dataset within a fixed computational budget.«
>
> Matthias Feurer et al., 2015, [FKE+15]

The authors formalised this in Definition 2.3 as:

$$\textit{For } i = 1, ..., n+m, \textit{ let } x_i \in \mathbb{R}^d \textit{denote a feature vector and } y_i \in Y \textit{ the corresponding target value. Given a training dataset } D_{train} = \{(x_1, y_1), ..., (x_n, y_n)\} \textit{ and the feature vectors } x_{n+1}, ..., x_{n+m} \textit{ of a test dataset } D_{test} = \{(x_{n+1}, y_{n+1}), ..., (x_{n+m}, y_{n+m})\} \textit{ drawn from the same underlying data distribution, as well as a resource budget } b \textit{ and a loss metric } \mathcal{L}(\cdot,\cdot) \textit{, the AutoML problem is to (automatically) produce test set predictions } \hat{y}_{n+1}, ..., \hat{y}_{n+m}. \textit{ The loss of a solution } \hat{y}_{n+1}, ..., \hat{y}_{n+m} \textit{ to the AutoML problem is given by } \frac{1}{m}\sum_{j=1}^{m} \mathcal{L}(\hat{y}_{n+j}, y_{n+j}). \tag{2.3}$$

Feurer et al. further state that AutoML may be viewed as a *combined algorithm selection and hyperparameter optimisation* (CASH) problem. CASH was first introduced by Thornton et al. as a combination of the two problems of first selecting a suitable algorithm for a learning task and then optimising the hyperparameters associated with it [THHLB13]. The formal definition by Thornton et al. is provided in Definition 2.4:

$$A^*_{\lambda^*} \in \underset{A^{(j)} \in \mathrm{A}, \lambda \in \Lambda^{(j)}}{argmin} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A^{(j)}_{\lambda}, D^{(i)}_{train}, D^{(i)}_{valid}) \tag{2.4}$$

A denotes a set of algorithms while Λ represents the associated hyperparameter spaces. The goal is to find the joint algorithm and hyperparameter setting that minimises the loss on a dataset D over K cross-validation folds. CASH is sometimes also referred to as the *full model selection* problem [EMS09] [HKV19, p. 5]. However, when viewed from an online training perspective, the definition provided in Equation 2.4 is not sufficient as it does assume finite datasets whereas online training is centred around continuous data streams. Hence, based on [FKE+15, Imb20], Kulbach et al. define the *online CASH problem* as stated in Definition 2.5 [KMB+22]:

Let $\mathcal{A} = \{A^{(1)}, ..., A^{(R)}\}$ be a set of step independent algorithms, and let the hyperparameters of each algorithm $A^{(j)}$ have a domain $\Lambda^{(j)}$. Further, let $S = e_1, e_2, ..., e_t, ...$ be an ordered sequence of examples of possibly infinite length and let t be the current observed example. Further, let $S^- = e_0, ..., e_t$ be an ordered sequence of past examples. Each example $e_i = \{x_i, y_i\}$ is a tuple of p predictive attributes $x_i = (x_i, ..., x_{i,p})$ and the corresponding label y_i. Let $\mathcal{L}(\mathcal{P}_{g,\overrightarrow{A},\overrightarrow{\lambda}}(S^T), S^V)$ denote the loss that algorithm combination $P^{(j)}$ achieves on a subset of validation examples $S^V \subset S^-$ when trained on $S^T \subset S^-$ with hyperparameters $\overrightarrow{\lambda}$. Denote that $S^T \cap S^V = \emptyset$. Then the Online CASH problem is to find the joint algorithm combination and hyperparmeter setting that minimizes the loss:

$$g^*, \overrightarrow{A^*}, \overrightarrow{\lambda^*} \in \underset{p^{(j)} \in \mathcal{P}, \lambda \in \Lambda^{(j)}, A \in \mathcal{A}, g \in G}{argmin} \mathcal{L}(\mathcal{P}_{g,\overrightarrow{A},\overrightarrow{\lambda}}(S^T), S^V) \quad (2.5)$$

In order to solve the CASH problem within the context of AutoML, Hutter et al. describe three central methods, namely HPO [HKV19, p. 3-33], *meta-learning* [HKV19, p. 35-61], and *neural architecture search* [HKV19, p. 63-77] (NAS), which are briefly outlined in the following.

Hyperparameter Optimisation

Hyperparameters are parameters tied to the specific ML algorithm, as opposed to the model, and are set before training commences. Every ML algorithm is affected by hyperparameters which directly impact its performance. An example for a hyperparameter is the learning rate α, which governs the step size in the popular *gradient descent* algorithm [CZ13, p. 131-133]. A low value might increase model quality at the cost of longer training time while a high value facilitates faster convergence with the possibility of the algorithm failing to find an optimal or near-optima solution.

Finding these values for such hyperparameters is usually referred to as HPO or hyperparameter tuning and is often a labour-intensive manual trial-and-error-process. In the context of AutoML, it is essential that HPO is conducted in an automated fashion [HKV19, p. 3-4]. The formalisation of this optimisation problem is given by Hutter et al. in Definition 2.6 [HKV19, p. 5].

$$\lambda^* = \underset{\lambda \in \Lambda}{argmin}\ E_{(D_{train}, D_{valid}) \sim D} V(L, A_\lambda, D_{train}, D_{valid}) \tag{2.6}$$

For an algorithm A operating on a dataset D, HPO tries to find a combination of hyperparameters λ which minimises the loss L [HKV19, p. 5]. In practice however, aside from the loss, multiple optimisation goals are often considered [HB16]. This may include conflicting objectives, such as model complexity versus training accuracy [Ige05].

To address the problem of HPO, several methods exist. Hutter et al. compiled a summary of state of the art approaches [HKV19, p. 7-18]. The authors broadly divide them into two categories: *blackbox HPO* [HKV19, p. 7-13] and *multi-fidelity optimisation* [HKV19, p. 14-18]. Blackbox methods are sub-divided further into *model-free optimisation* [HKV19, p. 7-8] and *bayesian optimisation* (BO) [HKV19, p. 9-13]. Model-free methods entail the well-known *grid search*, sometimes also referred to as *full factorial design* [Mon13, p. 183-232], where sets of values for each hyperparameter are defined and then organised in a grid, which is then searched exhaustively for the best combination. Grid search is computationally expensive as it scales exponentially with the dimensionality of the search space. Hence *random search* poses an alternative where hyperparameter combinations are sampled at random until a satisfying solution is found or a computational or time budget is exceeded [BB12]. In cases where the impact of the different hyperparameters differs from one another, this works better than grid search [HKV19, p. 7]. Figure 2.11 illustrates a simplified example. Random search is able to evaluate more unique values for the individual parameters while requiring the same amount of trials as grid search does.

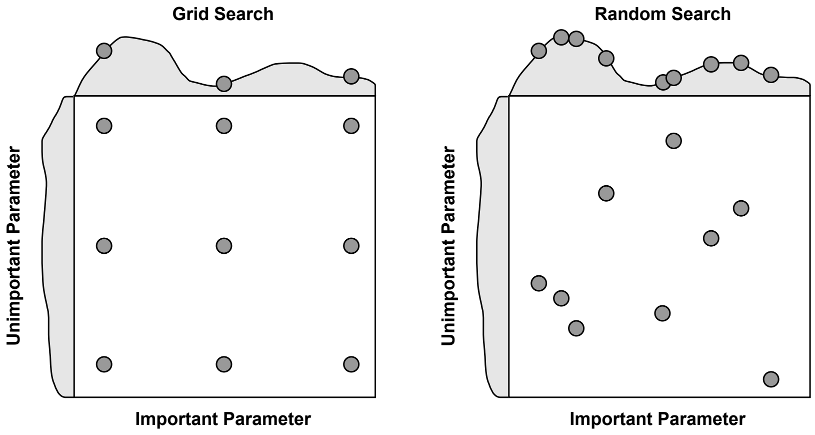

Figure 2.11: Comparing grid and random search over nine trials of minimising a function with two parameters, one important and one unimportant. Based on [BB12].

However, both grid and random search scale poorly compared to more advanced methods. Hutter et al. note several more model-free approaches, namely standard population-based methods such as *evolutionary algorithms* or *particle swarm optimisation* [HKV19, p. 8]. Especially the *covariance matrix adaption evolutionary strategy* [LH16] is cited as both simple and competitive within the frame of blackbox optimisation [BBO18].

Another popular technique is the aforementioned BO [HKV19, p. 9-13]. As the name suggests, BO is leveraging the *Bayes' theorem* [Joy21] insofar that it utilises prior information to navigate the search space in order to find the extrema more efficiently. It does so by defining a probabilistic *surrogate model*, sometimes also referred to as *surrogate function* or *response surface*, which is a Bayesian approximation of the objective function. It is vital that the surrogate model can be evaluated efficiently [SSW$^+$16]. Based on prior knowledge, the *acquisition function* then selects the next best sample within the search space for evaluation. In turn, the evaluating the sample triggers an update of the surrogate model. This process is repeated until the optimum or at least a satisfying solution is found, or until computational resources are depleted [BCdF10].

In contrast to the described blackbox approaches to HPO, multi-fidelity optimisation poses an alternative, especially for large data sets where exploring different hyperparameter configurations is costly and time-consuming. At its core, it tries to scale down the learning problem, optimise the hyperparameters over this simplified problem, and then re-apply the results back to the original problem [HKV19, p. 14-18]. This can be achieved through various techniques. In its simplest form, reducing the overall data by only using small representative subsets for training and HPO is sufficient to achieve satisfying performance while saving training time [Pet00, VDB04, KPB15, SST16]. A more advanced technique is *learning curve-based prediction for early stopping*, which, based on few training iterations, extrapolates the learning curve for a given hyperparameter configuration and automatically terminates the training when the predicted learning curve looks unpromising [DSH15]. Furthermore, *bandit-based algorithm selection methods* exist which are concerned with choosing the most promising algorithm and hyperparameter configurations from finite sets. Popular algorithms include but are not limited to *sequential halving* [KKS13] and *Hyperband* [LJD$^+$17]. Figure 2.12 illustrates the bandit-based approach of sequential halving (by different authors also referred to as successive halving).

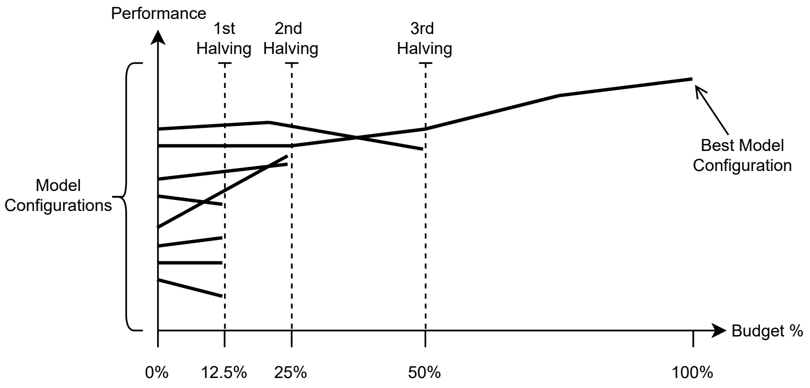

Figure 2.12: Sketch of sequential halving with eight model configurations and three halvings. Based on [KKS13].

The algorithm functions based on the assumption that high performing configurations already present themselves in early stages of training and continue to perform well over the entire training data. Starting from a large pool of model configurations, each is fed portions of the data until enough computational budget is spent for a cut-off to occur. In this, the lower half of model configurations on the performance curve is discarded. Training then continues with the remaining half until the next cut-off point is reached and so on, until only the best performing single model configuration remains [KKS13].

Noteworthy in this regard is also the relatively recent emergence of evolution-based algorithms for online CASH, such as *EvO AutoML* by Kulbach et al., which utilises a genetic programming approach [KMB+22]. Note that different optimisation techniques are not necessarily mutually exclusive. For example, *BOHB* combines BO with Hyperband in order to ensure both high performance and fast convergence [FKH18]. Lastly, algorithms exist which are able to choose fidelities adaptively at runtime. For example, *multi-task Bayesian optimisation* is able to dynamically switch between low- and high-fidelity based on a given acquisition function, allowing for cheaper exploration in the beginning of the optimisation process versus costly exploitation of promising hyperparameter configurations towards the end [SSA13].

Meta-Learning

Meta-learning describes the process of *learning to how to learn* [HKV19, p. 35]. In machine learning, this means applying experience gained on previous learning tasks to the problem at hand. To do so, meta-learning exploits information about prior models and model evaluations as well as the task properties [Van18]. Learning from prior models subsumes several popular techniques. In *transfer learning*, models trained on some task are used as a basis for training efforts on a new but *similar* task [TP98]. Another technique which gained traction in recent years, especially in the context of deep learning, is *few-shot learning*. Compared to transfer learning, few-shot learning is able to produce capable models with very little training data. In that, it tries to mimic humans in their ability to learn new tasks using prior experience and few examples only [LUTG17].

Neural Architecture Search

NAS is an AutoML method targeted at training *artificial neural networks* (ANN), especially in the context of deep learning. As the name suggests, NAS primarily deals with engineering neural architectures in an automated fashion [HKV19, p. 64]. It does so across three optimisation dimensions. The *search space* dimension characterises all theoretically possible architectures and is parameterised by the number of layers, the operations these layers conduct as well as the hyperparameters associated with these operations. Techniques exist to limit the potentially infinite search spaces beforehand. This includes simple measures, like limiting the maximum number of layers, but also encompasses more advanced methods, such as crafting architectures by stacking pre-defined cells (building blocks containing sub-architectures) [ZVSL18]. The second dimension *search strategy* defines the methods to systematically explore the search space. To do so, it leverages the entire portfolio of optimisation techniques. This includes random or grid search as well as BO, heuristics like evolutionary algorithms, gradient-based approaches, and finally, meta-machine-learning in the form of reinforcement learning agents optimising ANN architectures. The third dimension, the *performance estimation strategy*, entails methods to speed up measuring the performance of different architecture and parameter combinations as examining many training-validation cycles is computationally expensive. These methods include but are not limited to reducing training time by using less data or fewer training epochs in order to generate low-fidelity performance estimates, extrapolating learning curves of models trained on reduced data, warm-starting training processes based on already trained models, and training one-shot models which can be viewed as a supergraph from which subgraph models are derived which, in turn, inherit the once-trained weights of the supergraph [EMH19].

Solutions

Based on the theoretical foundations established above, a multitude of different AutoML solutions has emerged within recent years. Arguably the first one gaining widespread recognition was *Auto-WEKA* by Thornton et al. [THHLB13]. It is based on the popular *WEKA* data mining and machine learning framework [FHW16]. In its latest iteration, it utilises BO to achieve efficient HPO.

Since the early days of AutoML, numerous other solutions have entered the market, with a few examples being provided in the following. *TPOT* by Randal S. Olson et al. [OBUM16] is built on top of scikit-learn [PVG+11] and is centred around a tree-based pipeline and hyperparameter optimiser. It utilises genetic programming to optimise entire machine learning pipelines. *Auto-Keras* by Haifeng Jin et al. [JSH19] is an AutoML framework for the development of artificial neural networks, primarily providing NAS functionality using Bayesian optimisation. It is built on top of the widely used *TensorFlow* [AAB+15] API *Keras* [Cho15]. *H2O.ai* by Erin LeDell and Sebastien Poirier [LP20] provides a sophisticated AutoML suite for both conventional tasks and deep learning. It covers pre-processing, training, HPO using random search, and benchmarking of models. H2O.ai can be interfaced with through various APIs as well as a web-frontend. Thus, it is one of the more professionalised AutoML frameworks aimed at providing a full-scale solution for the industry end-user. Similar, however cloud-based, services are also provided by Big Tech, namely *Amazon SageMaker Autopilot* [Ama22], *Google Cloud AutoML* [Goo22], and *Microsoft Azure Automated Machine Learning* [Mic22]. In addition to solutions marketing themselves as strictly AutoML, implementations of many of the above-mentioned techniques can also be found in more traditional ML solutions. An interesting framework to name here is *River* by Montiel et al. which provides online learning as well as HPO techniques specifically tailored to it [MHM+21].

Discussion & Ongoing Research

The previous sections characterised AutoML alongside the three central methods HPO, meta-learning, and NAS as laid out by Hutter et al. [HKV19]. This, in its scope limited, understanding of AutoML almost exclusively addresses the in Definition 2.4 explained CASH problem. However, the ML development pipeline entails more steps outside of model training and optimisation. In 2021, He et al. [HZC21] reviewed the state of the art on AutoML from this more holistic view and uncovered several ongoing research gaps. Among others, they note that more application areas need to be explored as many studies focus on computer vision tasks. The authors also argue that interpretability and robustness, e.g. dealing with imperfect real-world or even adversarial data, is an open issue.

He et al. further stress that AutoML solutions do not sufficiently cover the entire machine learning pipeline, but instead tend to tightly focus on the model training and optimisation aspect. Lastly, lifelong learning, e.g. in the form of online training, is not provided by most current generation AutoML solutions. Figure 2.13 summarises the findings as an overview of the machine learning development and deployment pipeline, roughly highlighting to which extent activities are supported by available AutoML solutions.

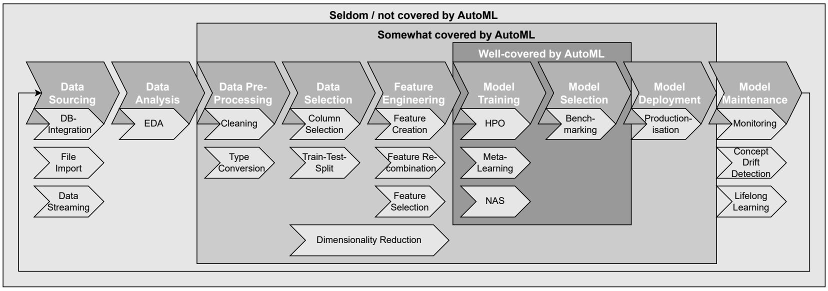

Figure 2.13: The ML development and deployment pipeline as supported by current generation AutoML approaches. Based on and expanded upon [HZC21].

Closing this section, an end-to-end solution to automate, or at least assist the end user in the activities alongside, the ML development and deployment pipeline is needed for AutoML to unfold its full potential. Regardless, the current generation AutoML solutions provide enough sophisticated functionality for this thesis to build upon and benefit from.

2.2.5 Machine Learning Engineering

In an effort to professionalise the development, deployment, and use of ML models, starting from the 1990s several process models for ML engineering have been conceived. One of the most popular ones to this day is the *Cross Industry Standard Process for Data Mining* (CRISP-DM) [WH00]. The process model defines six phases in the life cycle of a data mining project, which are illustrated in Figure 2.14.

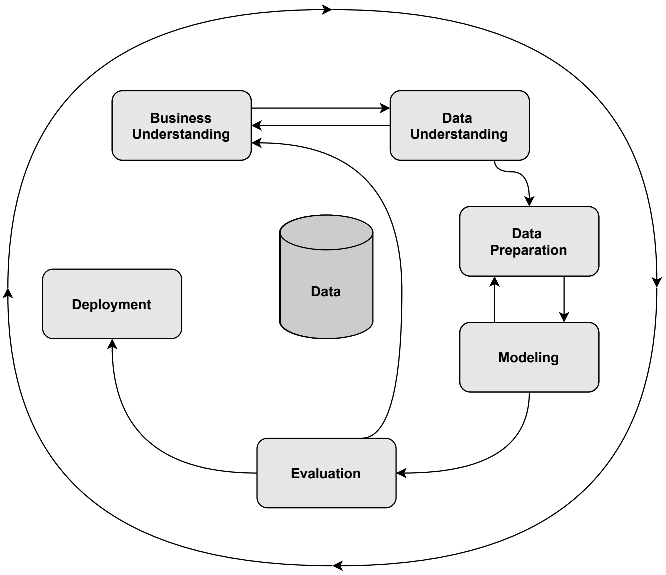

Figure 2.14: Phases of the CRISP-DM reference model by [CCK$^+$00, p. 13].

As shown, the phases are naturally arranged in an iterative way. Moving back and forth depending of the outcome of a particular phase is deeply engrained in the process model. These six phases are described in detail in the CRISP-DM data mining guide by Chapman et al. [CCK$^+$00]:

1. *Business understanding* makes up the first phase and starts with defining the business needs and requirements. This is then translated into a data mining problem and completed by establishing a rough plan on how to proceed.

2. *Data understanding* describes the next phase in which data is collected and analysed. The goal is to familiarise with the data, establish a bearing on its quality, and potentially form early hypotheses on hidden information relevant to the problem. Data and business understanding benefit each other and therefore, bouncing back and forth between the two phases is expected and encouraged.

3. *Data preparation* as a third phase deals with compiling the data set from the raw data. This includes cleaning, transforming, and selecting subsets of data.

4. *Modelling* is the subsequent phase in which the predictive model is built. In the context of machine learning this would include algorithm selection, training, and HPO. As this is a highly exploratory phase, changes to the underlying dataset occur frequently. Hence, a back and forth between data preparation and modelling is expected.

5. *Evaluation* denotes the fifth phase and dictates a reflection on the model performance against the business needs and requirements. At the end of this phase, a decision has to be made on whether to actually use the model in production. If the model proves insufficient, backtracking to the first phase of business understanding is possible in order to start the process anew.

6. *Deployment* marks the last phase and generally aims at productionalising the predictive model to be used in its target environment, fulfilling the business needs and requirements. In a more generalised context, this extends to the presentation of learnings generated from the data to key stakeholders.

In the decades to follow, researchers sought to improve and built upon CRISP-DM in various ways [AAH08, MPCOF+17, ALAMM+18]. Until recently, the focus remained on the development of predictive models. However, with the advent of the process model *Machine Learning for Production* [EFH+20] and the *Process Model for AI Systems Engineering* [HSP+21], the application area broadened and became more holistic, supporting the entire AI life cycle and aiming at both engineering and operating end-to-end AI solutions. The change of perspective includes emphases on customer integration, handover, and operation as well as maintenance of AI systems. This thesis remains firmly grounded in the tried and trusted process CRISP-DM prescribes yet at the same time lends itself to the operation and maintenance focus of the more recent process models.

2.3 Chapter Summary

This chapter laid out the managerial and technical foundations for this thesis. In Section 2.1, production was defined and different types and organisational approaches were distinguished. Classification schemes for production types were used to highlight the MTO focus of this thesis. Five models of production organisation were discussed an the job shop production model was highlighted as most relevant for this thesis. Concluding the production section, the concept, goals, and system landscape of PPC were introduced. Section 2.2 described the foundations of AI as relevant to this thesis. Several definitions of AI were provided before settling for ML as the main AI-application domain. This was briefly defined and further systemised by highlighting both the types of learning, with supervised learning identified as the most relevant in the context of this work, as well as introducing applicable learning techniques, such as online learning. Considerations were given on different feature engineering techniques for both offline and online learning. Then the problem of concept drift in ML was introduced, the four archetypical concept drift patterns were discussed, and various applicable detection and adaptation methods were presented. With the ML basics covered, AutoML was introduced as a means to automate steps of the laborious ML engineering process. Its central methods HPO, meta-learning, and NAS were explained before mentioning existing solutions and discussing the application potentials of AutoML in this thesis. The section was concluded by a brief overview of process models for ML engineering which serve as a framework for the method developed in this thesis.

3 State of the Art

Towards Machine Learning for Lead Time Prediction

Moving on from the fundamentals introduced in the previous chapter, the following zeroes in on research directly relevant to this thesis. This more narrow view on the subject is produced by a structured literature search and subsequent review focusing on the state of the art regarding ML-supported methods for lead time prediction (LTP). Starting with an overview of two distinct approaches to formulate the problem in ML-terms, relevant works tackling it are introduced in chronological order. The final sections summarise, compare, and contextualise these works resulting in the highlighting of several research gaps.

3.1 Machine-Learning-Supported Methods for Lead Time Prediction

LTP can be formulated as both a regression as well as a classification problem. Figure 3.1 illustrates the different problem formulations. When viewed as a regression, the target variable lead time (LT) is seen as continuous. This allows for more accurate predictions, given the model performance is high enough. Contrary to this, a classification approach represents LT as a discrete set of classes. Each class defines an interval on a continuous time scale.

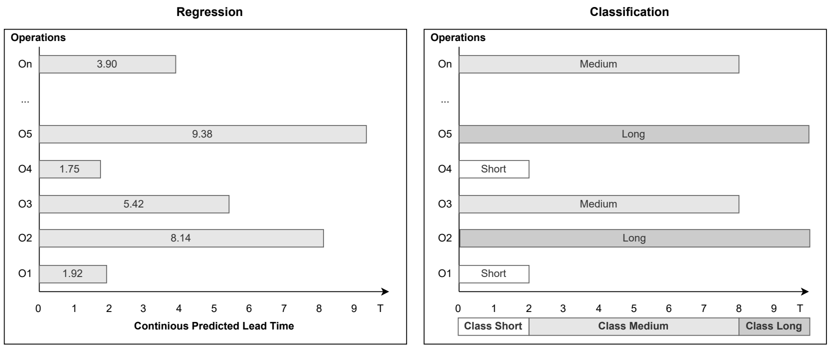

Figure 3.1: LTP as both regression and classification problem.

As seen in Figure 3.1, three classes, *Short*, *Medium*, and *Long*, are defined. The class *Short* ranges from zero to two (exclusive) time units, *Medium* from two to eight (exclusive) time units, and *Long* from eight time units onwards. This approach has the distinct advantage of being easily interpretable. Especially when exact predictions are not required, it provides a good ballpark estimate for how long an operation is going to take. However, determining the number of classes and defining their intervals is highly dependant on the problem space. The literature knows examples for both regression and classification approaches, with regression being the most discussed of the two, which are highlighted in the following sections.

3.1.1 Scope & Structure of the Literature Review

In order to keep the review within the scope of this thesis, works specifically targeting highly automated mass production are omitted. This mostly affects works on predicting cycle time in semi-conductor manufacturing in highly automated environments, unless these works carry some aspect particularly relevant to this thesis. In some papers, it is unclear whether the authors attempt to predict overall LT or just processing time (PT). Hence these are subsumed under one section.

In addition, there is a body of work, driven mostly by a group around Günther Schuh, on *transition time prediction* (TTP) as a subset of LTP. These papers are clearly separated from the main body of work on LTP and *processing time prediction* (PTP) for a better understanding.

3.1.2 Lead & Processing Time Prediction

The literature on LTP and PTP is further divided into the above described regression and classification approaches. Works on both are introduced in the following.

Regression Approaches

As early as 2003, ML was first applied to estimate LT [RW03]. Raymaakers & Weijters predicted LTs in batch production found in the food, chemical, and pharmaceutical industries. The authors randomly generated job data similar to real-world examples. They then trained and tested *regression models* (RM) and *artificial neural networks* (ANN) on the data. AANs outperformed the RMs due to their ability to fit to complex non-linear relations.

Further research was conducted by Öztürk et al. [ÖKÖ06]. The authors worked on LTP for make-to-order (MTO) production in a job shop environment. They modelled three distinct scenarios, i.e. shop floor layouts, with six machines each as well as ten different types of products which are produced in each of the scenarios. The authors used simulation to generate training and test data from their scenarios. They then used *Cubist* [Rul03], a tool to create rule-based prediction models, to train and evaluate *regression trees* (RT) on the simulation data. A key aspect of their work is an elaborate feature selection method based on the assumption that the number of occurrences of an attribute in a Cubist ruleset is a strong indicator of its predictive power. The authors developed a *weighted attribute usage ratio* (WAUR), which is expressed in Equation 3.1 as:

$$WAUR_i = \frac{\sum_{j=1}^{N} x_{ij} w_j}{\sum_{j=1}^{N} w_j} \tag{3.1}$$

j notates the Cubist rule index and N is the total number of rules in the ruleset. w is the number of occurrences which match the rule j. x is one if attribute i is used in rule j and zero otherwise. This ratio between zero and one is computed for every feature out of the 26 features they used. A threshold is defined and features with a WAUR below the threshold are removed from the model. The authors tried different thresholds between 0.0 and 0.9 and then compared model performances using the *mean squared error* (MSE) metric. They have found that feature elimination based on the WAUR slightly improved the predictive quality of a model up to a problem-specific threshold of about 0.5 to 0.7, when the MSE begins to rise again sharply.

Using *discrete event simulation* (DES), Alenezi et al. simulated three different multi-resource, multi-product, MTO shop floor scenarios: small, medium, and large [AMT08]. The small scenario was comprised of three product types and three production resources (i.e. machines). The medium scenario covered three products and ten production resources. And the large scenario encompassed 10 product types and 20 production resources. The authors trained and evaluated the four algorithms *support vector machine / regression* (SVM / SVR), *exponential smoothing* (ES), *moving average* (MA), and ANN. They furthermore experimented with different parameters such as kernels for SVMs, smoothing constants for ES models, window sizes for MA models, and layer architectures for AANs. Aside from the classic *root mean squared error* (RMSE) as a performance metric, the authors also used the *mean absolute percent error* (MAPE) to express relative performance differences. As a result, the authors found SVMs using a linear kernel and ε-insensitive loss function performing the best. In addition, they see a need for the development of more adaptive models which generalise better over time in future research.

The group around de Cos Juez had access to a data set covering the production of 524 batches of metallic components for the aerospace industry [dCGMT10]. It consisted of 12 features: batch size, CNC machining time, grinding time, milling time, heat treatment time, surface treatment time, horizontal latheing time, vertical latheing time, inspection test time, individual serial number available (yes / no), raw material cost, and forecasted manufacturing cost. The authors trained an SVM in order to predict overall LT. In addition, they trained another model based on a *Cox-Regression* [Cox72] which outputted importance values for the 12 features. Given this information, the authors proceeded to optimise the model performance by removing eight of the twelve features.

They highlight that the additional information about the feature importances is a powerful tool to better understand the factors involved in LTP.

In [MDF+14], the authors based their work on a real-world moulding company which offers *engineer-to-order* (ETO) products. The group were able to utilise detailed job data which was mostly of categorical nature and contained 10 features: number of cavities, type of hardening, side of injection, mould size, core cap, number of ejector rings, temper evident (yes / no), type of data, surface quality, and number of basic components. The authors also had access to recorded LTs on an operation basis. Contrary to most ML-based approaches in this section, the authors opted for a *case-based-reasoner* (CBR) in conjunction with a similarity measure which is based on the *Euclidean distance* between the features of the different jobs. They compiled a database of completed jobs (cases). So when a new job arrives, this job is compared to all other jobs in the database using the similarity measure. The CBR then interpolates the LT of the new job based on the most similar already completed job. Finally, the new job is stored in the database as a new case. In an example job, the authors were able to achieve a highly accurate prediction which was off by just 3.15%, or about 40 hours in total. However, they do not provide more general metrics on the overall performance of their approach.

Another simulation-based approach was conceived by Li et al. [LYWF15]. The authors aimed to predict LTs for a batch production semi-conductor manufacturing system containing 10 stations. They defined a multi-stage method which is based on a comprehensive DES model. This model is used to identify strong predictor variables and then perform *design of experiments* in order to provide simulation data on the predictor variables. This simulation data is then used to derive statistical RMs for LTP. An emphasis was put on considering variables describing the shop floor state, i.e. job arrival times, production load, and resource availability, as well as the interdependencies of multiple concurrent jobs competing for a limited amount of production resources. Using the the semi-conductor simulation model, the authors compared their approach to another solely simulation-based method by Hopp and Sturgis [HS00]. [LYWF15] showed that their own method outperformed the method by Hopp and Sturgis across four out of six performance metrics.

Mori and Mahalec based their work on a real-world use-case in steel plate production [MM15]. They describe the domain as one with high-variance low-volume products where steel plates are made to order tailored to a specific application. Hence PPS, and by extent also LTP, are problems depending on a multitude of variables. The authors therefore defined a multi-stage approach centred around *Bayesian networks* (BN) in conjunction with *decision trees* (DT), allowing them to derive the network structure from historical data. Using this network, they were able to predict the probability distributions of production loads and production times. In order to validate their approach, they used the available data to define two test cases. In the first case, they tested their model under the circumstance that only variables associated with customer demands are known. In the second test case, they also made information about operating conditions available to the model. The authors found that their BN model performed better in the second test case, i.e. when more features were available, but was also satisfying the first test case. They emphasise that their single-model approach covering both production loads and production times is easier to maintain than having multiple models for the different variables. However, they stress that more features regarding the shop floor state, such as the amount of in-process inventory, the number of workers, and the failure rate of each machine, could improve model performance.

An approach combining simulation and ML was conceived by a group around Pfeiffer and Gyulai [PGKM16]. The authors built a simulation model for wafer fabrication in a small-sized parallel flow shop system based on MES log data using DES. They then used the simulation data to train and compare the different algorithms RT, *linear regression* (LR), and *random forest* (RF). RFs performed best, however, the authors highlighted that RFs are especially sensitive to range changes in the predictor variables. Hence, they also considered model performance degrading over time due to distribution shifts in the underlying predictor variables. Thus, the authors proposed active retraining on fresh data to prevent models from degrading, for example by using their simulation approach.

A second work by the group around Pfeiffer and Gyulai was based on a use-case covering the production of optical lenses for eyeglasses in a flow shop system [GPN$^+$18]. These lenses are highly customised products and are individually made to order. The authors used MES log data from the real-world production to compare different methods for LTP.

They started out with an analytical method based on *Little's Law* [Lit61] in both periodic and rolling horizon configurations and then ran that against the ML-based methods LR, RT, RF, and SVM / SVR. The analytical method based on Little's Law was outperformed by all other methods with RFs performing the best. However, the authors stress that LR is simpler and thus easier to interpret. They also found that periodic retraining was necessary to maintain model performance and thus should be tightly integrated with the digital twin.

In another work by the same group, the authors conducted a case study on a semi-conductor manufacturer whose shop floor was organised in a job shop manner [LGA+18]. Data from the MES as well as information about the machine status and the customers spanning over two years was provided to the authors. They compared the different regression algorithms LR, *ridge* (RR) and *lasso regression* (LasR), *multivariate adaptive regression* (MAR), SVM, *k nearest neighbour* (kNN), RT (bagged, unbagged, and boosted), RF, as well as ANN. Again, the RF performed best on this problem.

Finally, a fourth work by the group around Pfeiffer and Gyulai lays out the concept of a new production data analytics tool for LTP [GPBG18]. Based on a simulated case study around a flow shop production, the authors conceived a closed-loop production controller which, powered by ML, predicts LT in real time as well as suggesting decisions on job prioritisation. The system is also capable of periodically retraining the models. Figure 3.2 illustrates the proposed architecture.

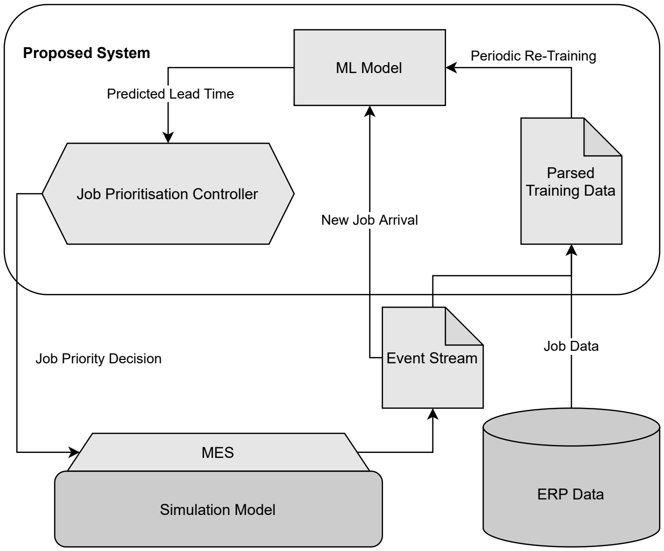

Figure 3.2: Architecture of the closed-loop production controller with real time LTP engine by [GPBG18].

The authors proposed an *event-driven architecture* in which the model for lead time prediction is fed an event, enriched with ERP data, for every new job arriving. Once the prediction is made, the job prioritisation controller then alters the job's priority accordingly and feeds this information back into the MES. The group tested their approach against a FIFO scheduler and a method based on Little's Law [Lit61] while measuring the total lateness of jobs with a seven day horizon. Their ML-based system outperformed the other two. The authors highlight that this event-driven approach with periodic retraining is capable of adapting to the dynamic changes on the shop floor.

Classification Approaches

As stated in Section 3.1, LTP can be treated as a classification problem. Only one work by Lim et al. was known at the time of writing in this regard [LYS19]. The authors used real-world data from different sources revolving around the production of customisable electronics test and measurement equipment. After merging the data, the authors removed outliers and performed feature engineering as well as encoding resulting in a total of 81 features. Later, they applied recursive feature elimination and ended up with eight features used in their model. In the following, they discretised the recorded continuos LTs into eight different classes marking the LT in days. Finally, the authors trained an SVM and compared the model to an RF and an ANN. The SVM model performed best at an F-Score of 0.852. However, it exhibited longer training and inference times than the other algorithms. The group highlights that, aside from the actual LTP, their classification approach also allows the easy identification of products with critically long LT.

3.1.3 Transition Time Prediction

Aside from authors presenting methods for predicting entire LTs, there are two groups explicitly attempting the prediction of TT. This covers the timespan the product is waiting or being transported between two individual operations and thus makes up a major part of the overall LT. As to how large that contribution is, studies by a group around Schuh et al. provide various estimates. They argued that up to 99% of overall LT are made up of TT [SPSF19]. In later works this was lowered to 95% [SPM+19] and up to 90% [SGST20] respectively. However, an exemplary case study showed that, when accounted for planned off-time in shift schedules, TT only made up 41.3% of the overall LT [Sau20, p. 187]. A further 43.4% was made up by these planned off-times and the remaining 15.3% were PTs. Still, methodising TTP remains highly relevant.

Starting out as a work on understanding the factors influencing cycle time (there defined as an accumulation of processing, inspection, transport, and other waiting times) for a use-case in semi-conductor manufacturing, Meidan et al. identified the waiting time as the most variable factor in their particular scenario [MLRH11]. They designed a simulation model based on the production of flash memory chips and generated data on 123 simulation scenarios.

Applying pre-processing and filtering using *conditional mutual information maximization* to the data, the authors brought down the number of features from 182 to 50. They then applied a *selective naïve Bayes classifier* (SNBC) in order to predict one of the three discrete waiting time classes. They compared their method regarding its accuracy against an ANN, a *multinomial logistic regression* (MLR), and a DT. The SNBC put forward by the authors scored comparably to the other methods though did not outperform them. As a more important result, the authors note that out of the top 20 factors impacting any given waiting time, eight are related to previous operations.

The group around Schuh et al. published a series of papers on predicting job-specific TT. In their first work, the authors started out from a simulation-based but real-world use-case in the aerospace sector [SPLS18]. In their scenario, they have identified TT, especially waiting times, as the main driver of LT. Arguing from this position, the authors presented a basic three step approach to determine job-specific TTs:

1. *Identification of influence factors*, in which a company-specific set of influence factors is derived from the raw data and then classified.

2. *Analysis of influence factors*, in which the influence factors are analysed and relationships between them are uncovered.

3. *Determination of output format*, in which TTs are predicted and fed back into the planning process.

Their second work was based a real-world use-case in the machining equipment sector [SPSF19]. They applied an RT and found that, compared to the static TTs set in the company's ERP system, their approach was 61.0% to 73.2% more accurate.

In parallel, the authors more closely investigated factors impacting TTP [SPM$^+$19]. Based on the method for TT classification presented by Meidan et al. [MLRH11], the authors conceived a method for production staff to identify key features with high impact on TT based on historical data. The concept merges domain knowledge of the production staff with the available hard data and was validated in a learning factory.

This method was further refined into a regression approach where the authors compared the performance across four scenarios using real-world data [SPH+19]. Scenarios I and II were dealing with raw data while III and IV were comprised of the raw data enriched by expert knowledge. Furthermore, in scenarios II and IV the models were subjected to hyperparameter optimisation (HPO). Scenario IV, i.e. the combination of a data set enriched by expert knowledge and HPO of the model, showed the best results. However, they do note that introducing expert knowledge into the predictive model impacts the planning which in turn impacts the prediction again. Hence, a control loop of prediction and planning needs to be engineered carefully.

In a subsequent work, the authors did take a closer look at this proposed control loop [SHPS19]. They identified three success factors for the accurate prediction of TT:

- *Data adequacy*, which they describe as the availability, integrity, and actuality of data.
- *Feature estimation*, which enables the forecast of future values of highly dynamic features at the time of prediction.
- *Measurement of the prediction performance and error allocation*, which prevent model performance from degrading over time.

Around these success factors, the authors designed a two-fold closed control loop for TTP with integrated model monitoring and retraining. This cascading close loop model for production planning and control (PPC) is illustrated in Figure 3.3.

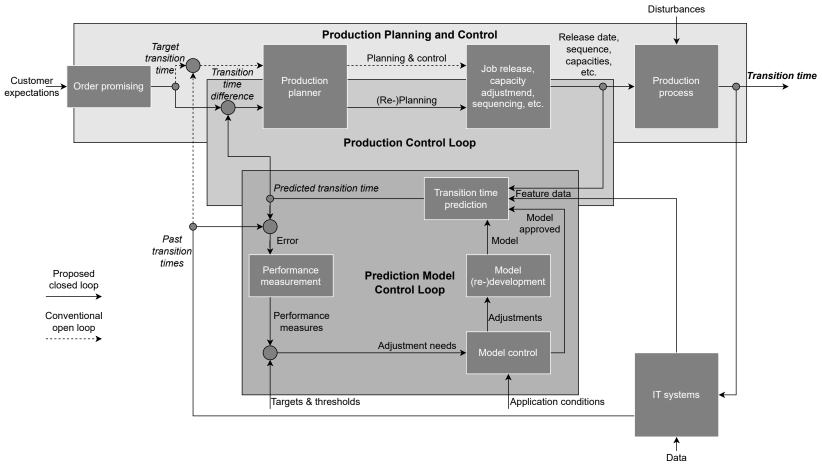

Figure 3.3: Cascading close loop model for PPC by Sauermann et al. [SHPS19].

The authors argue that in practice, planning activities are conducted in an open loop approach with TT being estimated unspecifically and based on historical averages. Thus they propose this integrated architecture of two cascading loops, one controlling the prediction itself and the other governing the monitoring and maintenance of the prediction model. They further refined their approach by presenting a holistic eight-step method for predicting job-specific transition times as shown in Figure 3.4 [SGS+20].

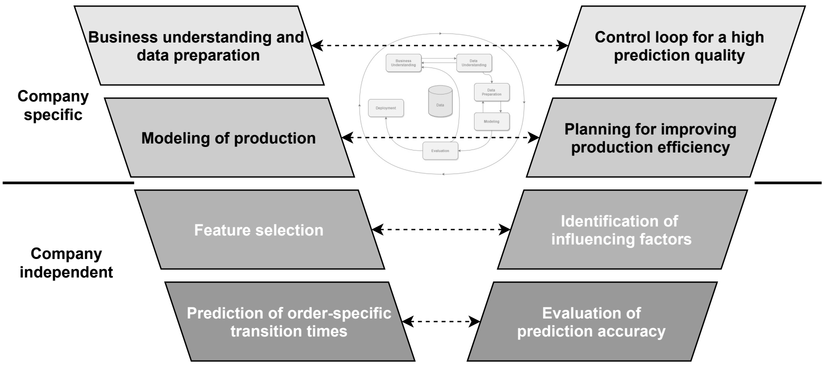
Business understanding and data preparation
Control loop for a high prediction quality
Company specific
Modeling of production
Planning for improving production efficiency
Feature selection
Identification of influencing factors
Company independent
Prediction of order-specific transition times
Evaluation of prediction accuracy
Business Understanding
Data Understanding
Data Preparation
Deployment
Data
Modeling
Evaluation

Figure 3.4: Methodology for databased prediction and planning of job-specific TT by Schuh et al. [SGS+20].

The method addresses both regression and classification tasks for TTP. The eight steps are described below:

1. Starting out based on the established CRISP-DM [WH00], the first step covers *business understanding and data preparation*. This includes analysing the application environment as well as ensuring the available data is complete, consistent, and free of errors.

2. Step two, *modelling of production and expert knowledge* is conducted in order to transform relevant relations from the application domain into machine-readable data.

3. In step three, *feature selection* is performed through a filter-wrapper approach similar to the one presented by Meidan et al. [MLRH11].

4. The *model training* follows in step four where hundreds of models with different combinations of filters, wrappers, and algorithms are trained in parallel. The authors provide several default filters, wrappers, and algorithms in their method but make clear that more can be integrated in the future.

5. In step five, *measuring performance* for the trained models is conducted for the evaluation.

6. Provided the models outperform expert predictions, *identifying influencing factors* for TTP is the focus of step six.
7. Step seven deals with the *integration of TTP into production planning processes*.
8. Finally, step eight governs the *establishment of the control loop*, mainly for steps three to seven.

As for steps 3 and 4, all combinations of filters, wrappers, and algorithms for both regression and classification as considered by the authors are illustrated in Figure 3.5.

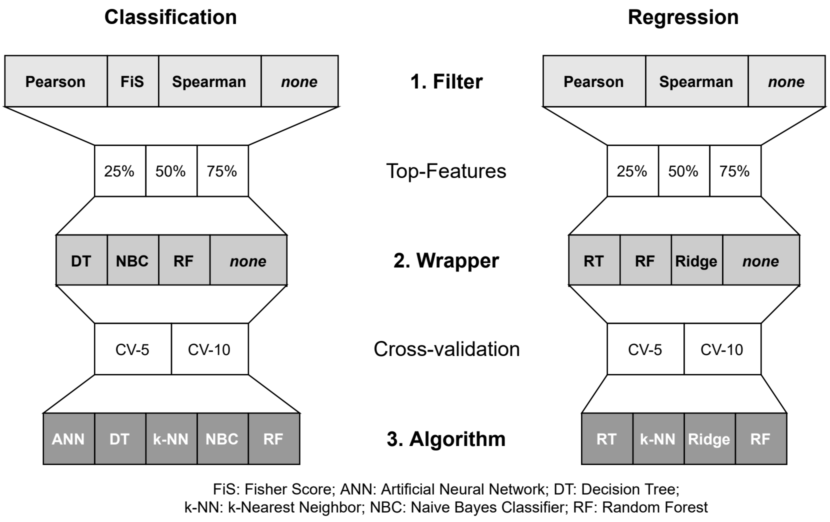

Figure 3.5: Tree of prediction model variants by Schuh et al. [SGS+20].

In order to benchmark their approach, the authors considered a real-world use-case in the machining equipment sector providing a dataset of over 13,000 jobs. They benchmarked both classification and regression approaches against the conventional manual predictions made by the production staff using a number of algorithms: ANN, DT, kNN, *naïve Bayes classifier* (NBC), RF, RT, and RR.

The highest accuracy of the classification approach was achieved by a DT at 29.8%, compared to 16.2% for the baseline manual prediction. As for the regression approach, the best *mean absolute error* (MAE) was achieved by an RF at 5.2 days, compared to 7.5 days for the manual prediction. The authors propose that more research is being done in order to improve the models as well as their integration into real-world system landscapes at manufacturing enterprises.

In a subsequent work by the authors, a real-world use case regarding the production of customised driving elements was investigated [SGST20]. The novelty of this work lies within the problem modelling in which the group attempted to perform TTP using a time-series data mining approach. A major challenge in this regard is transforming the available production data into a correctly ordered manufacturing time-series format. They argue that, from the available data sources (machines, operations, jobs), only the job-specific data should be considered as it is the only data which provides actual measured points in time as opposed to static master data from the other sources. Another challenge is aggregating this data in a meaningful way as any given point in time may occur only once in the data set while in reality on the shop floor, many actions occur concurrently. Some attributes, such as categorical data, are not suitable for aggregation and have to be left out. This aggregation furthermore inhibits job-specific TTP which the authors try to cope with by clustering the jobs beforehand and then predicting TT for the job type in question. The group them employed a *convolutional neural network* (CNN) to predict TTs on their time-series dataset. After manual hyperparameter optimisation (HPO), the model performed only slightly better than the conventionally planned TTs, with a MAPE of 53.89% for the CNN and 59.39% for the conventional planning. The authors concluded that, while their time-series approach is currently inferior to their other approaches, more research has to be conducted. Especially investigating other algorithms aside from a CNN as well as utilising automated HPO is needed. Furthermore, according to the authors, modelling the problem as a time sequence allowing for temporal data mining might yield better results.

Finally, the dissertation of Frederick Sauermann consolidates and expands upon the research by Schuh et al [Sau20]. The work unites the concepts laid out in the Figures 3.3, 3.4, and 3.5 and thus constructs a methodological basis to integrate TTP into planning and scheduling tasks. It furthermore expands on specific methods to preprocess relevant data, like cleaning and correction for known downtimes [Sau20, p. 126 ff.] as well as TT-specific feature engineering activities [Sau20, p. 130].

3.1.4 Literature Comparison

With the state of the art laid out in detail, key findings are summarised in the following. Tables 3.1 and 3.2 provide a compact overview.

Table 3.1: Literature comparison on LTP and PTP.

Reference	Production Domain(s)	Data Source	Problem Type	Algorithm(s)	Key Result(s)
[RW03]	Batch	Simulation	Regression	RM, AAN	AANs outperform RMs.
[ÖKÖ06]	MTO	Simulation	Regression	RTs	WAUR for feature selection.
[AMT08]	MTO	Simulation	Regression	SVM, ES, MA, AAN	Linear kernel SVMs perform best. Adaptive models needed.
[dCGMT10]	Batch	Real-World	Regression	SVM, Cox-Regression	Cox-Regression for feature importances.
[MDF+14]	ETO	Real-World	Regression	CBR	CBR & similarity measure perform well, emphasis on the use of product data.
[LYWF15]	Batch	Simulation	Regression	RM	Emphasis on the use of variables describing the shop floor state.
[MM15]	MTO, SS	Real-World	Regression	BN, DT	Prediction of probability distributions rather than discrete values.
[PGKM16]	MTO	Simulation	Regression	RT, LR, RF	RFs performed best, simulation-based retraining to prevent model degradation.
[GPN+18]	MTO	Real-World	Regression	Little's Law, LR, RT, RF, SVM	RFs performed best, tight integration with digital twin.
[LGA+18]	Batch	Real-World	Regression	LR, RR, LasR, MAR, SVM, kNN, RT, RF, ANN	RFs performed best.
[GPBG18]	MTO	Simulation	Regression	LL, Unspecified ML Algorithm	Event-driven architecture with periodic retraining.
[LYS19]	MTO, SS	Real-World	Classification	SVM, RF, ANN	Discretisation / classification of LT.

Table 3.2: Literature comparison on TTP.

Reference	Production Domain(s)	Data Source	Problem Type(s)	Algorithm(s)	Key Result(s)
[MLRH11]	Batch	Simulation	Classification	SNBC, ANN, MLR, DT	Waiting time prediction, high impact of previous operations.
[SPLS18]	Unknown	Simulation	Regression	-	High-level concept for TTP with ML.
[SPSF19]	Unknown	Real-World	Regression	RT	Validation TTP with ML.
[SPM+19]	Unknown	Real-World	Classification	SNBC	Identification of high-impact factors for TTP.
[SPH+19]	Unknown	Real-World	Regression	RT	Combination of expert knowledge and HPO.
[SHPS19]	-	-	-	-	Close control loop architecture.
[SGS+20]	Unknown	Real-World	Both	ANN, DT, kNN, NBC, RF, RT, RR	8-Step Method for TTP.
[SGST20]	MTO	Real-World	Regression	CNN	TTP as a time-series problem.
[Sau20]	MTO	Real-World	Both	ANN, DT, kNN, NBC, RF, RT, RR	Unification of [SHPS19] and [SGS+20].

As can be seen, most authors applied a regression approach using supervised learning. The data was sourced equally from simulations, usually using DES, and the real world. Most real-world data came in form of log data from MES or similar systems. Product data and data on the shop floor state were rarely considered. Most works focused on benchmarking different algorithms for LTP where model training and testing was done offline. Many authors noted that models do not generalise well enough on new data. Later works such as by Gyulai et al. started to consider online training, adaptive models as well as integrational aspects with the automation pyramid [GPBG18]. The group around Schuh et al. provides the most sophisticated body of work combining both algorithmic and architectural aspects into a more holistic method [SPLS18, SPSF19, SPM+19, SPH+19, SHPS19, SGS+20, SGST20].

However, they focus exclusively on TTP backed by expert knowledge. They are also the only group that considered a time-series approach, though with mixed results. The most comprehensive work on TTP was provided by Sauermann [Sau20]. While many authors sought to automate certain aspects of the ML engineering process, the utilisation of AutoML was not considered explicitly.

3.2 Research Gaps

Based on the review of the state of the art on ML methods for LTP, five research gaps are highlighted that lie within the scope of this thesis:

- *Unifying* the work done with regards to LTP as an overall goal, with PTP and TTP as crucial parts of it, into a *holistic approach* has not been achieved yet.
- The problem of *models generalising poorly* to new circumstances, i.e. handling concept drift, is often mentioned but rarely addressed adequately.
- Following the previous research gap, most authors considered an offline training approach. Little research has been done on *applying online training* in this particular problem domain.
- While complex schemes for model development have been conceived, the potential benefits of simplifying this undertaking by utilising *AutoML technologies*, especially in combination with an *online training approach*, have not been investigated yet.
- Lastly, *service-based model deployment and maintenance as well as integration* with existing system landscapes, especially at SME, is outside of the scope of most works.

3.3 Chapter Summary

This chapter provided an in-depth dissemination of the state of the art in ML-supported LTP. After further systematising the problem of LTP into regression and classification modelling approaches, it defined the scope and structure of the successive literature review. It was found that the existing body of work could be distinguished into papers on combined LTP and PTP as opposed to works focusing on TTP. As a trend, early works experimented with various ML algorithms and established the general feasibility of applying an ML approach to LTP. Later works however revolved more around ML engineering approaches and system architectures as well as integration into PPC. The studies were summarised and compared based on their content along the categories production domains (mostly Batch / MTO / SS), data sources (real-world / simulation), problem types (classification / regression), algorithms, and key results. Concluding the chapter, several research gaps were derived to guide the work presented in this thesis. These highlight the need for a holistic approach to LTP while addressing generally poor model generalisation and adaptation as well as simplifying the model development process, rendering it more accessible for SMEs.

4 A Method for AutoML-Supported Lead Time Prediction

How to Leverage the Machine Learning Potentials

Building on top of the research reviewed in the previous chapter, and especially setting a focus on the existing research gaps, the following introduces a method for AutoML-supported LTP. This chapter is starting with an overview of the method's technical objectives and preconditions, defining the framework in which the method can be operated. It is followed by a brief overview of the method itself and completed by detailed descriptions of every step along the way. Furthermore, a suggested system architecture is sketched out. Finally, the method is assessed based on the technical objectives and differentiated from the other approaches laid out in Chapter 3. Earlier iterations of this method were published in papers on prototyping ML-supported LTP using AutoML [BO21] and benchmarking several AutoML solutions on two real-world case studies [BTO22]. The method also builds on integrational aspects conceptualised in another co-authored paper [SKK$^+$23].

4.1 Framework

The applicability of the method is dependent on a set of assumptions and preconditions laid out in this section. First however, the technical objectives of the method, as in what is tried to be achieved, shall be clarified.

4.1.1 Technical Objectives

In Section 1.2.1 of this thesis, the MO was introduced as:

> MO: End-to-end methodisation of a PPR-specific LTP for SME from the MTO domain using AI.

As was further explained in that section, LT divides into PT and TT. Therefore, a model for LTP needs to predict two target variables: PT and TT. The prediction needs to be performed for each operation a job undergoes as its total LT (TLT) is the sum of all PTs and TTs of each operation, which was formalised in Equation 1.1. For the remainder, note that an operation is considered as the realisation of a process. As was described in Section 3.1, the target variables can be modelled as either continuos, resulting in a regression problem, or discrete as a set of classes, resulting in a classification problem. If not stated otherwise, the method should therefore function the same way for both approaches. Figure 4.1 highlights the model's target variables within this framework.

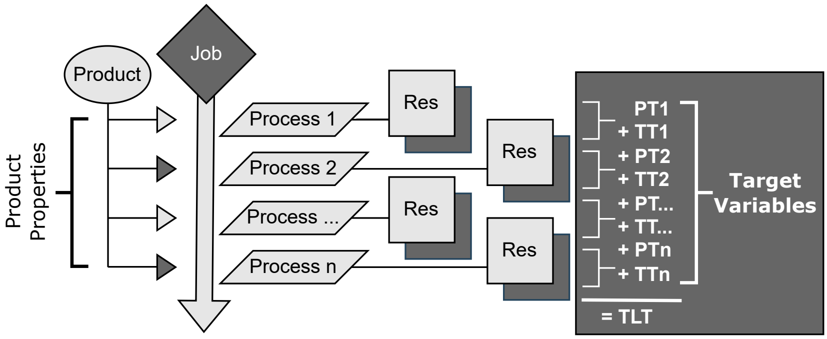

Figure 4.1: Target variables of the LTP model.

Furthermore, the LTP model is to learn from data using AI, supported by AutoML techniques, in order to build-up sufficient predictive capacities. It should also be equally integrateable and maintainable within an SME's system landscape.

Integrateable in the sense that it can be deployed, inferenced, monitored, and updated within the company network. And maintainable as in able to be updated in order to adjust to changing circumstances on the shop floor, signalled through the detection of concept drift in the input data. Therefore, the *technical objectives* (TOs) of this method are formally defined as follows:

> TO1: Enable LTP by utilising AutoML to develop models capable of predicting PPR-specific PTs and TTs.
>
> TO2: Enable integration of the LTP models into a system landscape where they can be deployed, inferenced, monitored, and updated.
>
> TO3: Enable the LTP models to adjust to concept drifts in the input data in order to maintain a constant performance.

4.1.2 Streaming Scenario

The method is aimed to function within a streaming usage scenario. In opposition to a static scenario, where data is being collected in a database and then processed batch-wise, the streaming scenario assumes a constant flow from various data sources with data being processed one-by-one once it becomes available. This approach is chosen as it is closer to reality where data emerges from different sources continuously, asynchronously, and independently of one another. Figure 4.2 visualises the difference. These two scenarios also coincide with the offline and online learning types, explained in Section 2.2.1.

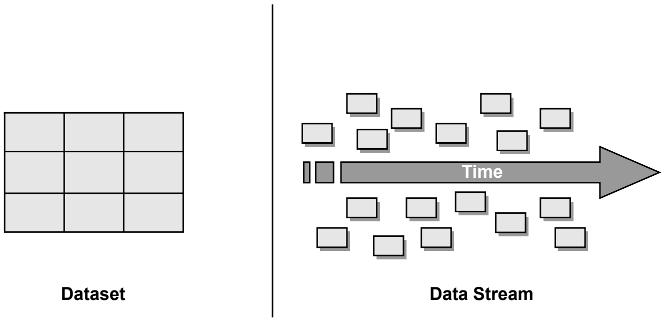

Figure 4.2: Finite static dataset versus infinite dynamic data stream.

The streaming scenario furthermore assumes that the data is not stationary, i.e. changes to its distribution or other properties may occur at any moment in time. If the distributional changes negatively affect a predictive model's performance, concept drift is present [GŽB+14]. An online learning approach can leverage these circumstances by monitoring the data stream for concept drift and adjust itself automatically once it occurs in order to maintain a high performance. This adaptability is a major advantage over the static offline approach.

In the context of a shop floor, the data stream is characterised by the different data points being generated independently of one another on the shop floor and within the surrounding company's systems. This can be anything from new jobs arriving, to feedback from work stations, up to sensor data being recorded. An example for the non-stationarity of a shop floor environment could be some upgrade of the machinery resulting in lower PTs, in turn affecting the performance of a model attempting to predict these. If the predictive model is able to detect the distributional change early, it can adapt to the new circumstances and therefore maintain its performance.

4.1.3 Preconditions

Several preconditions need to be fulfilled in order to successfully apply the method. These are built on and expanded upon the success factors for TTP as laid out by Sauermann et al. [SHPS19].

- As with any approach centred around ML, *access to data* is paramount. Viable data sources are described in Section 4.2.1. Not all data needs to be available up front. The method is designed to function incrementally and iteratively. Thus, acquiring the necessary data is part of the process. However, some form of information system needs to be present and accessible within the company.
- Naturally, management overseeing the method application are required to *commit the necessary resources* to the undertaking. This is in terms of both personnel, especially skilled labour from the areas of production planning and the shop floor, and computational resources, with access to the information system as well as computing power.
- Especially in the early stages of development, the *availability of experts* is crucial. This refers to inner-company experts, be it managers, planners, specialists, or similar. Their resources need to be committed to the model development for it to succeed as some steps benefit greatly from expert insight and assessment.
- Not a strict necessity but useful nonetheless is the *availability of baseline estimations* for the target variables. These may be manual predictions by production planners which can be used to establish a performance baseline to benchmark the model against. If no baseline exists, performance targets can be defined and adjusted as needed.
- *Accurate shop floor reporting*, especially of actual PTs and TTs, is of utmost importance in order to maintain a high input data quality. Shop floor staff need to be sensitised accordingly and provided with the right tools to record accurate data.

Figure 4.3 summarises the above-stated success factors for the method application. With these in place, the next sections describe the method in detail.

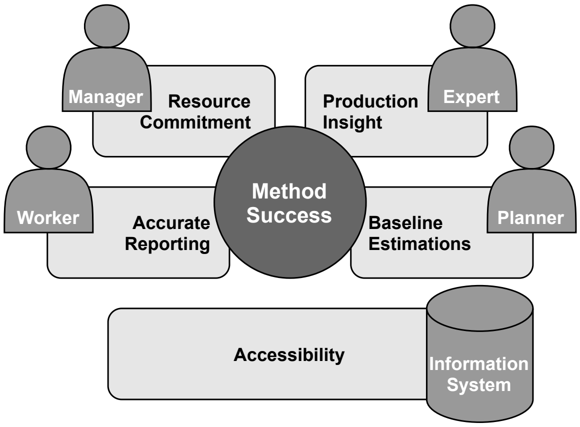

Figure 4.3: Preconditions for the success of the method.

4.2 Description

The method for AutoML-supported LTP presented in this thesis is made up of two stages with three blocks organising a total of twelve steps. The *prototype* stage covers the process which leads to a first complete LTP model iteration while the *production* stage is characterised by the use and adaptation of the model. The method's first block is revolving around the *data preparation*, covering the steps *sourcing*, *analysis*, *transformation*, *filtering*, *encoding*, and *selection*. This also includes tasks associated with feature engineering, which is not placed separately but instead is realised as part of the activities within the first block. The second block *model development* defines the three steps *configuration*, *training*, and *benchmarking*. Finally, the third block *model deployment* carries *integration*, *monitoring*, and *maintenance* as its associated steps. Blocks one and two mimic the traditional tried and tested approach of developing a predictive model. Block three is predicated on ideas from ML operations on how to provide access to these models in production and ensure they are maintained accordingly.

AutoML support is considered at every step within the limits of current generation AutoML approaches. Figure 4.4 provides an overview. Detailed explanations of each of the blocks and their respective steps is subject of the remainder of this section.

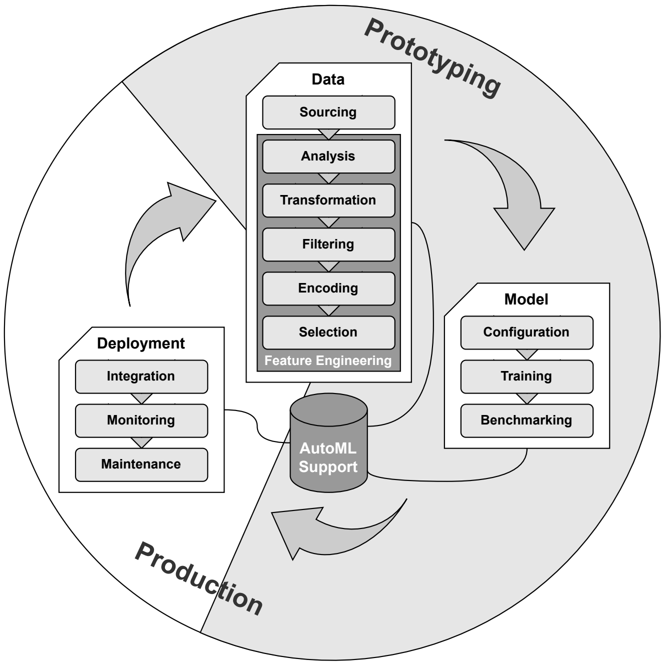

Figure 4.4: Overview of the method for AutoML-supported LTP.

Falling in line with ideas from software and ML engineering, represented in models such as the in Section 2.2.5 previously mentioned CRISP-DM [WH00], the method is following a state of the art iterative and incremental approach. While initially, the steps provide a logical sequential order to produce a first solution, development cycles can and should be repeated to increase and maintain high performance. Such, steps can be reverted to or repeated as needed and the method execution is never considered *done*. The main idea behind the iterative core of the method is providing a framework of constant self-improvement on both a managerial and technical level within a given company.

The role of AutoML is highlighted by the connections to the different blocks. As outlined in Section 2.2.4, current generation AutoML technology cannot support every step in the same capacity, and some steps still rely on human capabilities or technologies outside of AutoML. Future developments of AutoML technologies might lead to further AutoML substantiation within this method, however this goes beyond the scope of this thesis.

4.2.1 Data Preparation

Highlighted in Figure 4.5, the first block *data preparation* forms the starting point of the method. As the name states, the main object of interest in this block is the input data. The main goal is to obtain and shape the input data into a state so that it can be used for the development of a functioning predictive model. This block is therefore divided into six steps which are laid out in detail below.

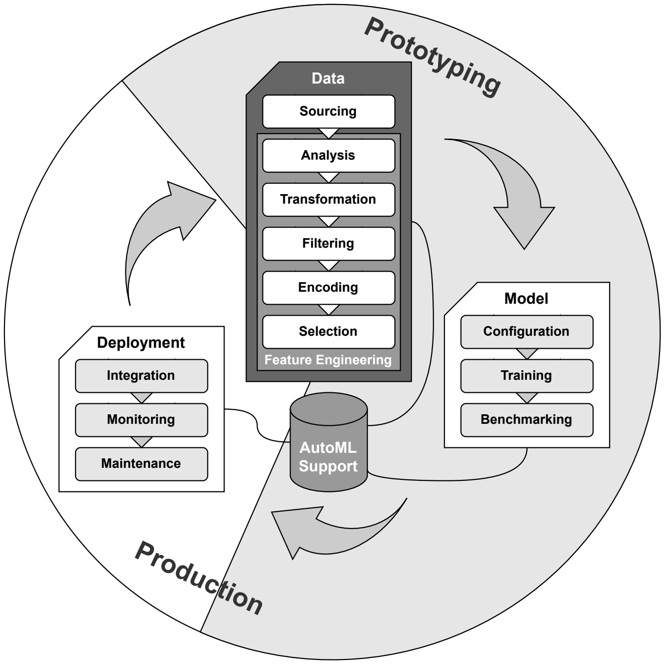

Figure 4.5: Block one of the method for AutoML-supported LTP: Data Preparation.

Sourcing

Sourcing suitable data poses the first challenge, especially for SME with heterogeneous system landscapes. Note that this is a highly individual task which needs to be tailored to the data sources accessible in the given company. Figure 4.6 provides an overview of potential inner-company data sources. Arguably the most important information can be obtained from job data. Following the PPR-paradigm, data on products, processes, and resources can be utilised as well. Finally, this data can be complemented further by expert knowledge. Of course, these data sources cannot be strictly divided as they are interlinked and share or build upon each other's information in order to operate the shop floor effectively. Note that some data can be considered (mostly) static master data, e.g. CAD data or resource specifications, while other is strong movement data, e.g. job data or shop floor state. From an ML perspective, it is also important to acknowledge which data is available a priori, in terms of at the time when the LTP is performed, and which data becomes available a posteriori, as in after the associated process has been completed on the shop floor and reported back to the system.

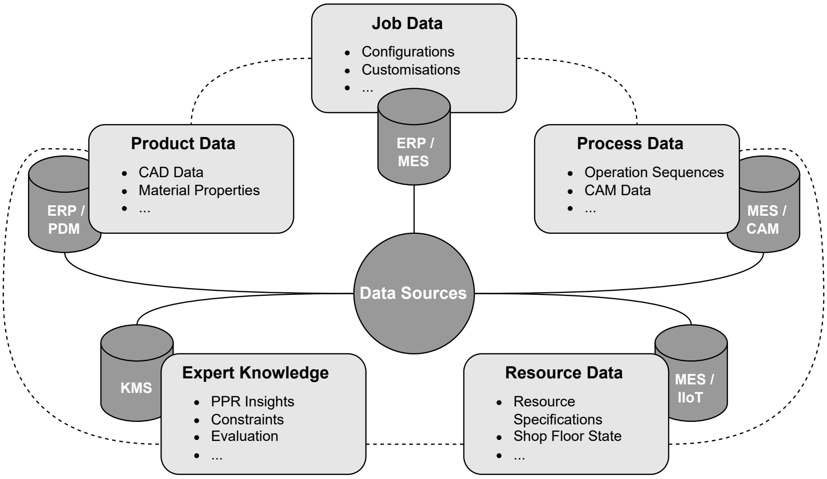

Figure 4.6: Potential inner-company data sources.

Starting on the top in the centre, *job data* is highlighted as the absolute minimum of information needed to enable the method. This data contains concrete customer orders, often including product configurations, lot sizes, customisations, and due dates but also activity logs and tracked progress. Note that this might also include the a posteriori target variables PT and TT. As described in Section 3.1.2, Meidan et al. already discovered that aggregated progress data, especially information about previous operations, has a positive impact on prediction quality [MLRH11]. Suitable systems to extract this data from are enterprise resource planning (ERP) or manufacturing execution systems (MES).

On the top left corner, *product data* makes up the next relevant portion of data. It encompasses information tied to the product model and is mostly made up of static master data. This ranges from CAD-models and drawings, through bills of materials, all the way down to concrete physical properties of the materials used in the products, e.g. the forming temperature of some type of steel. Product data can be obtained through ERP and product data management (PDM) or similar systems. As mentioned in Section 3.1.2, the importance of product data as a predictor for LTs was already recognised by Mourtzis et al. [MDF+14].

Moving on to the top right corner, *process data* contains information describing the processes producing the product. This can range from operation sequences to process parameters up to sophisticated computer-aided manufacturing (CAM)-data, containing detailed programmes. Suitable data sources are MES or CAM systems.

The lower right corner represents the *resource data*. Static master data, such as machine specifications and capabilities, come into play here. This is complemented by movement data including personnel schedules, resource availability, production load, and other shop floor circumstances, forming the combined *shop floor state* as a data representation of the actual reality on the shop floor at any given moment in time. Pointing back to the work by Li et al. mentioned in Section 3.1.2, variables describing the shop floor state show high predictive potential and can greatly influence lead time [LYWF15]. Resource data is obtained from MES or industrial internet of things (IIoT) platforms.

As the lower left corner represents, the aforementioned data sources should be complemented with *expert knowledge*. This encompasses mainly the know-how of human experts (production planners, product developers, shop floor staff) on the products, processes, resources, and overall operation of the shop floor. Experts may also provide valuable constraints not explicitly modelled in the systems and can help in the evaluation of predictive models generated by the method. However, digitising expert knowledge is difficult, suitable sources for structured data are knowledge management systems (KMS). Another promising and more straightforward approach to harness expert knowledge is to enrich raw data from the other sources through direct expert modelling, as proposed by Schuh et al. [SPH+19]. These authors also note that careful considerations should also be made regarding the adequacy of the data. Furthermore, experts can be asked to rank or select potentially promising data in order to identify features of high impact [SPM+19].

When assessing the adequacy of potential data and data sources for the method, Big Data approaches can be applied. There exists a loose characterisation scheme, based on the *n Vs* of data which originates back to the three Vs *data velocity*, *data volume*, and *data variety* formulated by Laney in 2001 [Lan01]. Note that this notation has been extended to four, five, six, and more Vs by various actors in different contexts and there is no standard scheme agreed upon in the scientific community. However, it is still useful to consider various aspects represented by these Vs. For this method, seven Vs stand out, as explained below. However, this assessment approach is not a strict recipe for application but rather highlights characteristics that should be considered when assessing potential data sources.

- **Volume** describes the size of the data or the amount of data points. It is not possible to define one strict threshold which must be met in order for a data source to become suitable. Rather, it should be considered individually whether the data is sufficient for its specific purpose. For example, mostly static master data of material properties might be perfectly sufficient in low volume, representing just as many materials as the given company processes. While movement data, like information on jobs, is required in larger volume in order to cover the full spectrum of operations and product variety in an MTO setting. As mentioned previously, domain experts can be queried to support this assessment.

- **Velocity** refers to the speed at which new data becomes available. As discussed before, this is crucial as ground truth for important predictors might become available late in the prediction and planning process or, in case of the target variables, even only some time after the prediction. This, in turn, affects monitoring and maintenance activities of the LTP model. It is also worth noting that a flow of data (data stream) is often encountered in reality. However, data streams in a production environment are not steady with a constant flow rate. Rather, data generation is tied to events on the shop floor or in the overall company which can occur at any moment in time and independently of one another. This includes for example new high-priority jobs being released causing re-planning of the entire schedule, but also reaches all the way down to progress reports from shop floor staff on individual jobs which arrive as operations are completed.
- **Variety** denotes the differentiation into unstructured, semi-structured, and structured data, which also covers considerations about data types. In order to successfully train well-performing LTP models, structured data is required. The availability of such is highly influenced by the sate of digitisation in the company. In general, the more processes are digitised, the more structured data is available since information systems such as ERP or MES usually rely on state of the art relational databases as their data backbones. It becomes more difficult when dealing with CAD data or CAM programmes as useful information needs to be extracted and transformed which can prove laborious. Extracting information from paper-based shop floor data such as job tickets or printed drawings with handwritten instructions is both laborious and prone to error. In such cases, digitisation of these processes should be prioritised in order to create more useful structured data sources.
- **Veracity** means the degree of trust in the data. I.e. whether it is of sufficient quality and accuracy, and if it is complete. Problems in this regard may occur with any data that is entered manually. This extends to manual input of static master data, such as information on the resources or process parameters, all the way to shop floor reporting being inaccurate, incomplete, or missing entirely. Principles and guidelines enforcing double-checks of existing data and on entering new data help mitigate these problems.

However, further digitisation of processes and reducing the need for human input should be considered regardless. In any case, a certain amount of erroneous data is to be expected and needs to be taken into account when developing and using the LTP model.

- **Value** is assessed in order to rank the data according to its usefulness for the LTP model. This is used to remove data sources with little value or expand on others that provide information strongly benefiting the LTP model. Again, expert knowledge can provide an orientation on the value of various data and its sources. The value should be re-assessed after every method iteration in order to identify the strongest predictors and maintain performance long-term.
- **Variability** refers to differences and inconsistencies between different data sources. This is likely to occur when working with multiple information systems. Considerations need to be made on how to harmonise and unify the data from different sources. A value-based selection helps to reduce the complexity of the problem by ignoring data sources that provide little value. However, some effort still remains in order to produce a structured, complete, and synchronised data stream composed from multiple data sources.
- **Volatility** is the last of the seven Vs considered in this method and is concerned with the lifecycle of the data. Specifically, the question of how long any one data source or data point is valid should be answered. As an example, changes on the shop floor or the company's products or processes can lead to concept drift in the data. If this concept drift is so substantial that the performance of the LTP model is affected, adapting it is required. In this case, the data from before the concept drift outlived its usefulness as it does not reflect the current situation on the shop floor any more. In this case, the old data needs to be ignored in subsequent training efforts as to not jeopardise the model's performance. A close monitoring of the data stream and model performance in conjunction with expert oversight help mitigate volatility issues.

On a purely technical level, information systems offer some form of interface to extract data with. In the best case, a system provides a sophisticated interface, such as a *representational state transfer* (REST-)API [Fie00], which allows for specific querying and the return of structured data.

Semi-automated, periodic file exports also present an option, although a less comfortable one. Direct access to the underlying databases is often discouraged by the system providers but stands as a last resort for automated data extraction. Otherwise, data has to be extracted manually which is laborious and prone to errors. AutoML-support for this step is limited. There exist AutoML solutions which are able to perform simple queries for data on remote databases or REST endpoints [LP20]. However, even when available, this functionality is generally not sophisticated. Most AutoML solutions either offer file imports (.CSV) or rely on third party data management solutions [The23].

Analysis

Following the sourcing of data, an *analysis* should be conducted. This step is based on the exploratory data analysis (EDA) briefly introduced in Section 2.2.2 [Tuk77]. For this to be successful, a sufficient data sample is needed. No clear statement can be made on how much data is *enough*. In manufacturing however, to catch seasonal effects, a year's worth of production data provides a good starting point. To familiarise with the data, basic information on the columns is collected. This includes data types and counts, especially of null or *not-a-number* (NaN-)values which indicate missing or corrupted data. A simple example is shown in Table 4.1.

Table 4.1: Example metrics to compute on a data set.

Column	Data Type	Count Total	Count NaN
Column 1	Int	99,562	1,692
Column ...	...	...	...
Column N	String	99,562	0

By reviewing the data with domain experts, columns may already be classified as numerical or categorical data. As for numerical data, statistics such as mean, standard deviation, min, max, and quartiles are computed, as shown in Table 4.2.

Table 4.2: Example statistics computed on a per-column level.

Statistic	Column 1	Column ...	Column N
Mean	7,811.36	...	-
Std	1,878.54	...	-
Min	3.65	...	-
25%	535.00	...	-
50%	1,994.00	...	-
75%	7,204.00	...	-
Max	1,249,697.00	...	-

For categorical columns, it is useful to compute the amount of unique values. In addition, the distribution of each unique value may be analysed and visualised to gain further insights. An example is presented in Figure 4.7.

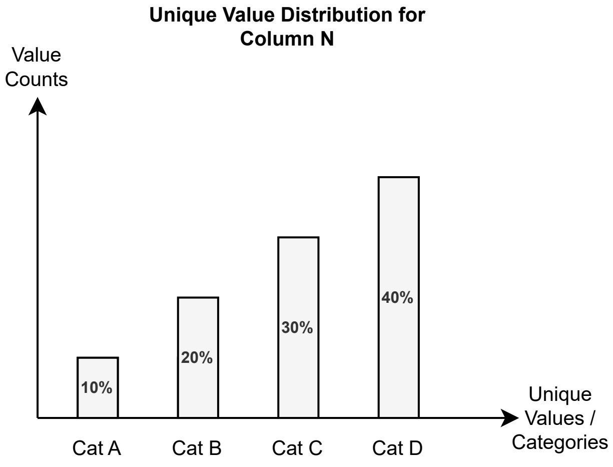

Figure 4.7: Example distribution analysis of unique values in a categorical column.

Especially interesting is information on processes to form an idea about what the most often executed, time consuming, or time variable processes are. A technique to further this understanding is applying histogram analysis on both an overall and individual process level in order to visualise the value distributions. This is shown exemplary in Figure 4.8.

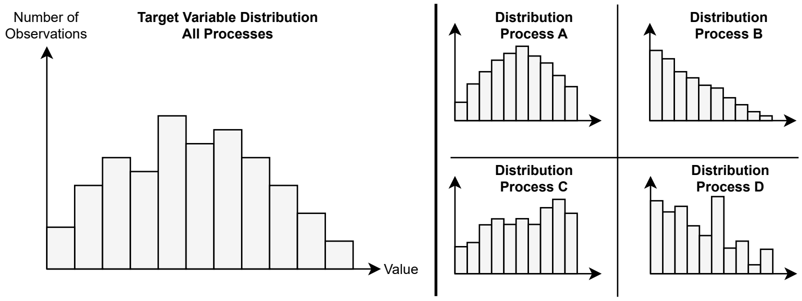

Figure 4.8: Example histogram analysis of both overall and individual process target variables.

Another established technique to specifically visualise locality, spread, and skewness of numerical data is box plotting. This can be applied to numerical columns in general but also to the target variable with regard to individual processes as illustrated in Figure 4.9.

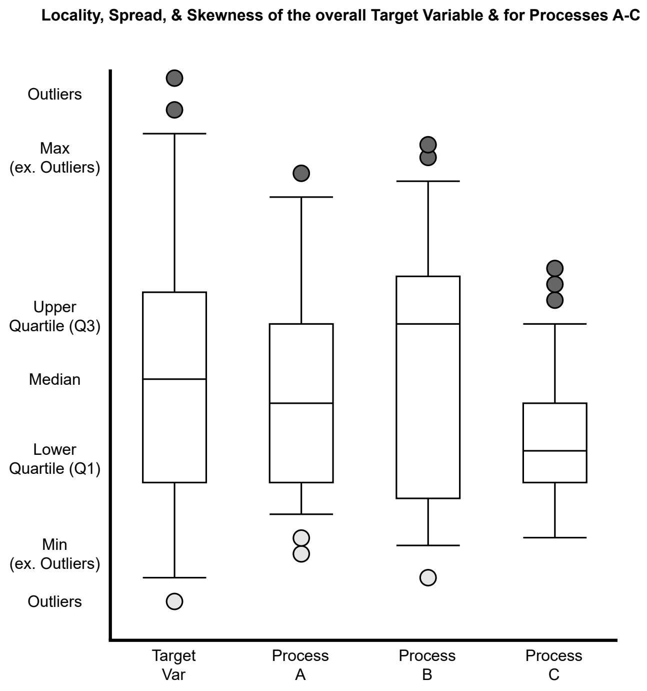

Figure 4.9: Example box plot analysis for four variables.

Further insight is gained by analysing columns outside of the target variable. This includes process parameters, such as temperature, but also extends to information on used materials or semi-finished products. Distribution analysis is applied to generate an overview about how processes are typically executed and what commonly used materials are.

As for the analysis of relationships between different columns, variables respectively, the correlation measures among all are calculated. This is visualised via a correlation matrix, highlighting strongly correlated variables, as shown in Figure 4.10. In general, this approach allows to explore the relationships between variables, thereby furthering the data understanding.

However more specifically, this becomes important downstream as some ML algorithms react more sensitively to correlated variables than others, potentially impacting performance.

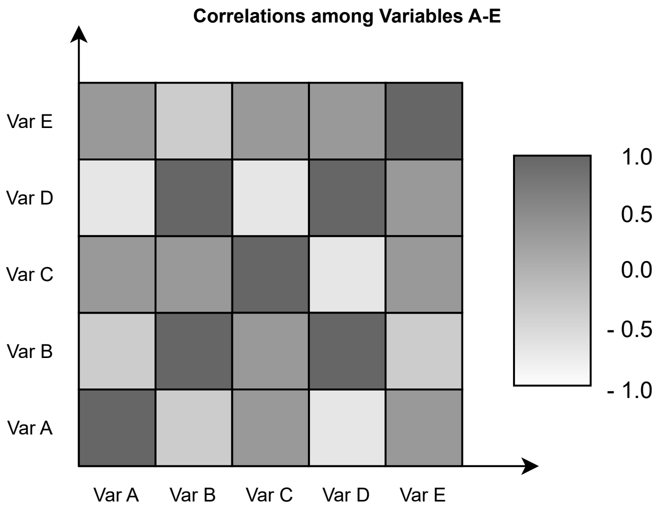

Figure 4.10: Example correlation matrix for five variables.

Aside from an aggregated correlation matrix, individual variables' relationships are visualised using a scatter plot. As an example, in Figure 4.11 the two target variables PT and TT are put against each other. Of course, this analysis may again be performed on an individual process level as well. Further recommended plot analysis includes plotting target variables over time, for example to visualise seasonal effects.

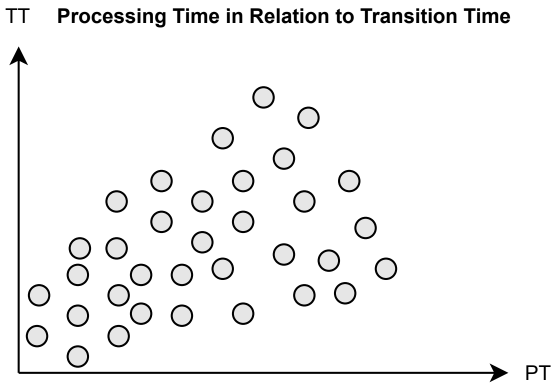

Figure 4.11: Example scatter plot exploring the relationship between the target variables.

Depending on the shape of the available data, further company-specific analysis may be performed as needed. As these analyses do not fall under the core AutoML techniques, support in this regard is non-existent. Some solutions are set up to automatically infer basic information such as mean values or NaN counts [LP20]. However, for actual analysis, third party data management frameworks [The23] in conjunction with visualisation tools [Hun07] provide the technical basis.

Transformation

The analysis provided the groundwork for the next step *transformation*. This serves as an umbrella term for several activities with the ultimate goal of producing a unified, structured, and machine-readable data stream as a basis for model development. Note that the activities described below are not following a strict order. Also, a complete rundown of all potential activities cannot be provided as this depends on the individual company and the circumstances it is operating under. Furthermore, the transformation extends to activities falling under the feature engineering. This governs feature generation, recombination, and substantiation.

As already discussed on data sourcing, not all data might be present in a digitised and structured form. Depending on the data source, it could thus be necessary to extract this data first. For example, in case of CAD- / CAM-files or similar artefacts, associated metadata or model-specific properties may be read and persisted in tabular form. Relevant handwritten data is digitised and, like other forms of manual data input, should be checked carefully. Considerations should be made on any handwritten data source as to what extent it is worth to transform it. Extracting already structured data from information systems is possible by querying APIs or databases. For use in data streams, database triggers or periodic polling present viable options so that recent data is always pushed into the stream as soon as it emerges.

Furthermore, the cleaning of errors in the data poses an important activity. This covers handling NaN values, either by removal or replacement. And it entails plausibility checks with subsequent handling of implausible data points. Incomplete or corrupted data should be identified and dealt with through completion or removal on a case-by-case basis. As with the data sourcing, experts can support this activity.

Another common technical activity are type conversions. This covers standard string-to-number or date-to-number operations but also extends to breaking up more complex data structures into arrays which are joined as columns into a table structure. On the subject of processing strings, other operations might be necessary. This includes breaking up strings which contain multiple pieces of information, as is often the case with manually inputted data. As for numerical data, be it integer or floating point, unit conversions are in order. Figure 4.12 illustrates some of the aforementioned transformation activities.

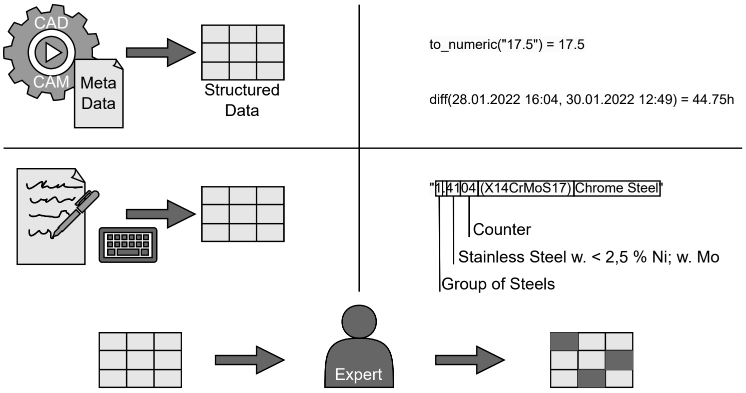
CAD
CAM
Meta Data
Structured Data
to_numeric("17.5") = 17.5
diff(28.01.2022 16:04, 30.01.2022 12:49) = 44.75h
"1.4104 (X14CrMoS17) Chrome Steel"
Counter
Stainless Steel w. < 2,5 % Ni; w. Mo
Group of Steels
Expert

Figure 4.12: Examples for data transformation activities with expert oversight. Clockwise from the top left to the bottom left: extraction of meta data from CAD / CAM, type conversion and recombination, string splitting and interpretation, digitising and structuring handwritten data.

Among the previous examples, strings were broken up to generate useful features. This is taken one step further by *substantiating* the feature. Considering the example situation shown in Figure 4.12 top right. In this case, the product material is represented as a string containing multiple pieces of information, most notably the material number according to DIN EN 10027-2:2015-07 [DIN15]. This number falls into three parts classifying a certain type of steel. However, even the broken-up string in this form is still *just* a categorical feature, providing limited informational value. Fortunately, by querying additional data sources outside of the company, like a data sheet on this particular steel, physical properties can be extracted, substantiating the feature and expanding on the informational value. The hypothesis behind this is that the physical properties of the material influence production processes which in turn affect PTs. Thus, by enriching the data with these substantiated features, stronger predictors are exposed. Substantiating the material feature only serves as an example meant to encourage creative thought when engineering features for the LTP model. Figure 4.13 illustrates this approach.

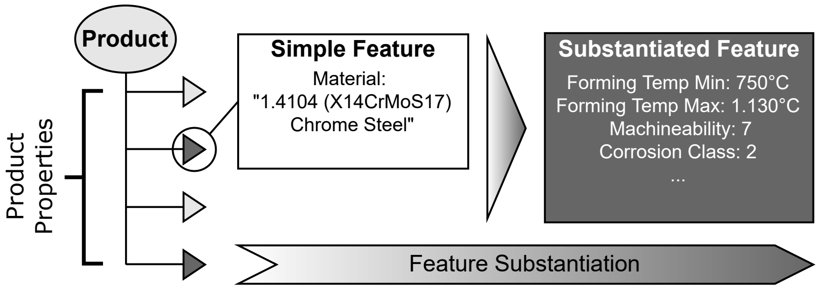

Figure 4.13: Example feature substantiation in which the material string label of a chrome steel with little informational value is transformed into a set of parameters representing the material's physical properties.

Special care should be applied to transformations on the target variables, PT and TT. First, a common unit should be defined which is easily interpretable. Depending on the company circumstances this may be minutes or hours but also days or weeks. Furthermore, it is crucial to harmonise LT with regard to the company's shift calendar. This is done by subtracting LT by the time which has passed outside of the regular working hours. This is important when a company does not operate 24/7 as the inclusion of irrelevant off-times, weekends, or holidays could negatively the LTP model's performance. For cases in which the working hours are not exactly known or vary in day to day operations, Sauermann proposed a heuristic to derive a shift calendar from the data [Sau20, p. 128]. This heuristic shift calendar is then applied to correct the recorded LT.

A special case of time data and often encountered in this production environment are timestamps. Note that these reference a single point in time and require interpretation or transformation in order to draw useful information from. One transformation is to split and extract certain repeating properties from a timestamp such as daytime, weekday, or month, in order to identify reoccurring relationships. In another approach, timestamps, i.e. points in time, are transformed into time spaces by calculating the distance between two timestamps, therefore putting them into relation. Analysing time spaces between job arrivals and releases serve as an example of this approach.

The above transformed data from different sources has to be merged in order to produce a cohesive data stream, as illustrated in Figure 4.14. The challenge in a data streaming scenario is synchronising the different data sources. This is achieved by buffering related events and only triggering the LTP model once a complete data vector is produced. In practice, hybrid systems are in order as static master data is seldom changing while movement data, such as job data, is entering the system often yet infrequently. Solutions to the problem of data merging will thus be highly individual to the company's system landscape.

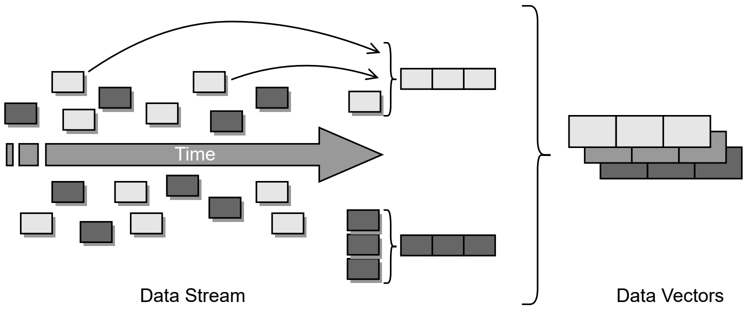

Figure 4.14: Merging of streamed data to data vectors for online learning.

After merging the data, as previously mentioned in Section 2.2.2, application of the data scaling operations standardisation [GE06] or normalisation [AG19, p. 40] should be considered carefully. Direct AutoML-support for this step is limited. Often encountered functionality extends to simple NaN or missing value handling; type recognition and conversion are also found in some solutions [LP20]. However, for more sophisticated feature engineering tasks, numerous third party tools have emerged in recent years. Some are focused on temporal and relational datasets [KV15], provide feature recombination to generate and evaluate new features [KSS16], or offer an entire collaboration platform to support the process of feature engineering [SWV17]. Thus, the transformation step is in part supported by such tools, yet it remains subject to close expert oversight and manual experimentation in order to produce a viable company-specific data stream for the successive steps.

Filtering

The next method step is *filtering*, which is closely linked to its predecessor transformation. The objective of this step is to remove unwanted data points from the data stream. Unwanted data comes in several forms and can be *irrelevant*, *implausible*, *incomplete*, or even *erroneous*. Irrelevant data means all data the LTP model does not benefit from. For example, data describing operations not directly affecting PPR-specific LT, such as maintenance, are filtered out. Implausible data has already been addressed briefly in the transformation step and covers data points with values which cannot be explained. If during transformation, these data points cannot be corrected, they should be removed. Similar is in order for incomplete data. In some cases, outright erroneous data might be encountered. Errors may occur due to technical reasons or, as is the case with manual inputs, human error. In the partially digitised system landscapes of SMEs, errors in the data have to be expected and handled accordingly.

Techniques to detect errors (and also implausibilities) can be subsumed under the term *outlier detection*, where, in its simplest form, the statistical distribution of a given variable is analysed to isolate data points situated far outside said distribution. Depending on the scenario, these data points are filtered out entirely or replaced by some value, for example an average. Outliers can indicate an error in the data but might also be valid and just the consequence of a rare event occurring. In an MTO scenario, such a rare event could be the execution of some seldom required laborious process resulting in an unusually high PT. Therefore, outliers should be detected and ideally reviewed by experts on a case-by-case basis. An assessment should be formed whether a learning algorithm could benefit from the outlier example or whether it should be removed in order to not potentially degrade the performance of the LTP model. Figure 4.15 illustrates the approach.

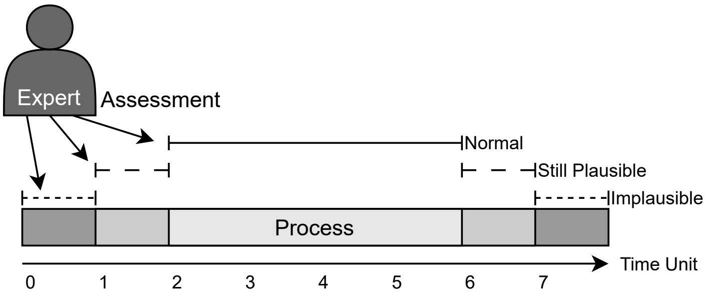

Figure 4.15: Expert assessing recorded processing times with regard to their plausibility.

As manual assessment is laborious, automating the identification of the right cut-off points for implausible values can be achieved through various techniques. An easy method for univariate outlier detection is based on the *Z-Score*, whereby the distribution of a variable is transformed to a 0 mean and 1 variance. An outlier is then defined by a threshold which is the product of the variance and a given factor, for example two or three times the variance. Thus, any value beyond this threshold is defined as an outlier.

Another simple method is based on the *interquartile range* (IQR). In this, the values of a variable (for example the target variable) are sorted by size and divided into four quartiles around the median (Q1 - Q4). These quartiles mark absolute points where 25%, 50%, 75%, and 100% of the variable's values fall below, with the median being positioned precisely on the second quartile at the 50th percentile. The IQR is then calculated by subtracting the first from the third quartile, as shown in Equation 4.1.

$$IQR = Q3 - Q1 \tag{4.1}$$

The IQR is then used to define a filter, which is produced by multiplying the IQR with some factor, thereby defining a range of acceptable values around the median. This is shown in Equation 4.2. Figuring out the best factor so that it filters out the extremes while retaining most of the useful data is the challenge with this approach and might require multiple iterations of experimentation.

$$Filter = IQR * Factor \tag{4.2}$$

The resulting IQR-filter is reapplied to the distribution in order to cut off values outside of the filter range. This results in extremely high and low values being removed from the data, as illustrated in Figure 4.16.

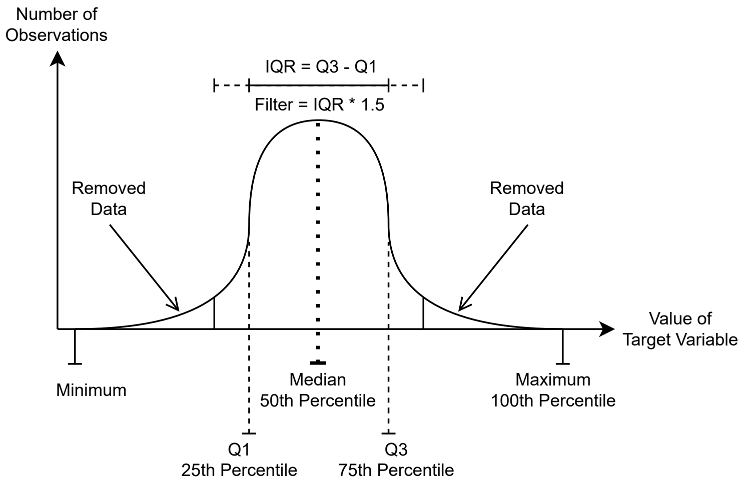

Figure 4.16: Application of an IQR-filter to the target variable with a factor of 1.5.

Filtering through the IQR is simple and easy to interpret but falls short when multivariate outliers or anomalies occur. In these complex cases, more sophisticated approaches are in order. One algorithm popular to this day is *DBSCAN*, originally developed by Ester et al. [EKSX96]. It presents a clustering functionality similar to the k-Means algorithm with the difference that the number of clusters does not need to be specified in advance. Subsequently, any data point that cannot be assigned to a cluster, is an outlier of some form. Another technique utilising deep learning is based on *autoencoders*. An autoencoder is a type of neural network that takes an input of data, *encodes* it into a simplified representation, and then attempts to *decode* it back to its original form [GBC16, p. 4].

However, to a degree there is always a *reconstruction error*, meaning more complex data might not be perfectly reconstructed in the decoding phase. This circumstance is exploited to detect data that does not fit the expected pattern by defining a threshold over the reconstruction error as a means to detect outliers. Figure 4.17 illustrates these two approaches.

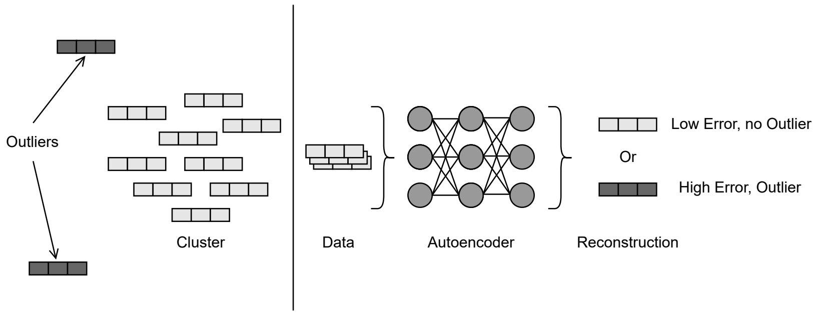

Figure 4.17: Identifying outliers with ML-based methods. On the left side, using a clustering approach such as DBSCAN [EKSX96]. And on the right side, using an autoencoder's reconstruction error [GBC16, p. 4].

Of course, the techniques mentioned here are only examples and the filter step is individual to the company's circumstances, requiring expert insights. As mentioned before, state of the art AutoML solutions can provide simple NaN-value filtering but fall short on more sophisticated techniques [LP20]. The statistical approaches via the Z-Score and the IQR shown above can be realised with little effort through third party libraries [The23]. For the advanced ML-based approaches, more sophisticated solutions for statistics or ML [PVG+11], and especially neural networks [Cho15], offer various algorithms.

Encoding

The next step in the first block is *encoding*. This means the transformation from categorical data, usually represented by strings, to integers. It is necessary as ML algorithms require numerical input data to learn from. In a production scenario, categorical data is often encountered. It may denote process IDs or materials, but can also stretch to information on customers or resources. In Section 2.2.2, two encoding approaches, label encoding and one-hot-encoding [ZC18, p. 78-79], where introduced. Figure 4.18 shows an exemplary application of these two alternatives where process names defined as strings are being encoded.

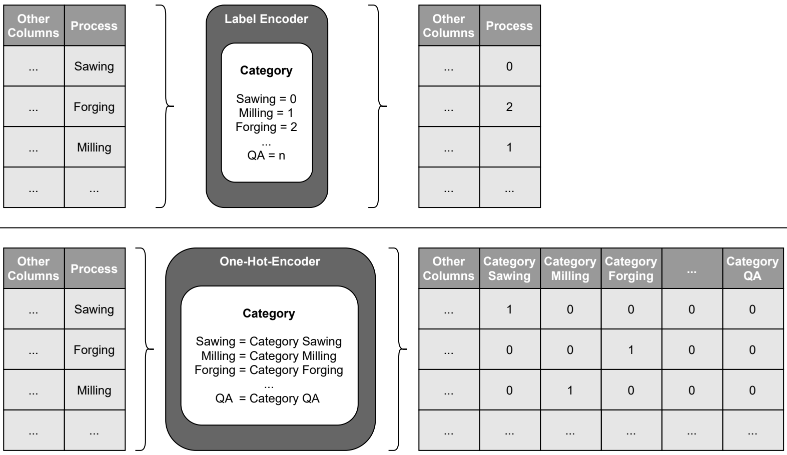

Figure 4.18: Exemplary application of a label encoder (top) and one-hot-encoder (bottom) [ZC18, p. 78-79].

The choice of encoder for the LTP model is a decision depending on the circumstances and especially on the available data. The aforementioned caveats of both approaches shown here, the potential misinterpretation of data as ordinal with the label encoder versus the curse of dimensionality with the one-hot-encoder [Bel03], have to be weighed against each other carefully. An iterative experimental approach is advisable.

Another thing to be aware of is that introducing new categories, for example due to changes on the shop floor, will result in the need to re-fit or extend existing encoders. Most AutoML solutions provide basic encoding out of the box. Some also attempt to guess whether a given column is of categorical nature [LP20]. However, a degree of expert oversight is still required to correctly handle categorical data.

Selection

The final step in the first method block is the *selection*. Contrary to an offline scenario, in which this would be conducted along the two axis of columns (features) and rows (test-train-splits), the online scenario only requires the selection of columns (features) as all data points passing the filter step are eventually fed the LTP model to learn from. As a starting point, all a posteriori features are removed from the data as they are by definition not available at the time of prediction. However, if deemed beneficial, these a posteriori features may be estimated through additional predictive models. According to Sauermann, considerations should include which features are worth estimating based on their influence, at which point in time they should be estimated in relation to the planning process, and which method is used to perform the estimation. Sauermann also notes that the effort of developing these auxiliary models should be minimised and simple statistical methods, such as linear regression, considered [Sau20, p. 131-132]. Figure 4.19 illustrates a simple selection example where the a priori available predictors are selected and complemented by an estimated delayed predictor not yet available at the time of prediction. Meanwhile, columns deemed irrelevant for the learning process are ignored.

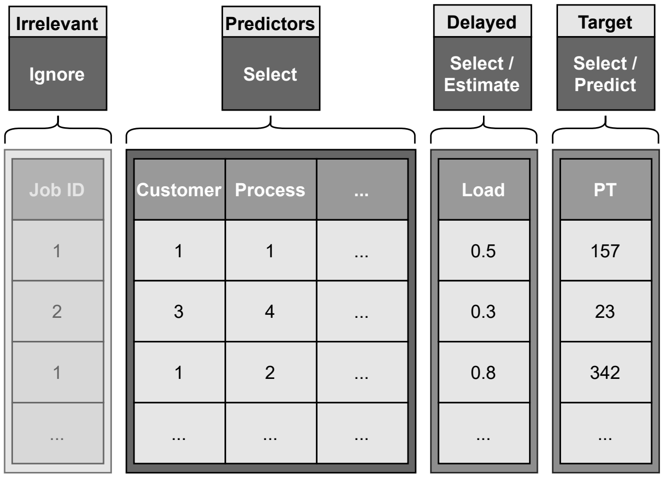

Figure 4.19: Exemplary selection decisions on selecting, ignoring, and estimating individual columns both for training and prediction at runtime.

In offline learning, the heavy lifting of the selection step would be performed by the filter, wrapper, and embedded approaches introduced in Section 2.2.2. To reiterate, filter methods perform statistical operations on the input data to compute filter thresholds, wrapper methods interact with the model to determine the selected features based on its performance, and embedded methods are similar to wrapper methods though are directly integrated into the learning algorithm [GE06]. As described in Section 3.1.3, Schuh et al. proposed a filter-wrapper-algorithm scheme for TTP [SGS+20]. However, as previously discussed in Section 2.2.2, these methods cannot be easily applied to an online scenario as all data needs to be available for the training process. As for filters, some can be adapted to function based on running statistical properties which does not require the entire data to be available upfront [MHM+21]. This, for example, extends to a filter based on the Pearson correlation which is also part of the proposed scheme by Schuh et al. For a more sophisticated approach tailored to online learning, the aforementioned Fast-OSFS [WYD+13] or SAOLA [YWDP14] algorithms can be applied. The development of a broader base of selection methods for online learning is however still a research gap of its own [GRB+19].

As for AutoML-support for the selection step, existing solutions often provide classic filter methods [OBUM16] or at least compute corresponding statistics, such as feature importances [LP20].

4.2.2 Model Development

With the data preparation complete, the second block *model development* commences as shown in Figure 4.20. The goal of this block is to use the prepared data to systematically find the best performing LTP model among a wide range of model configurations. To do so, three important steps have to be taken which are explained in the following.

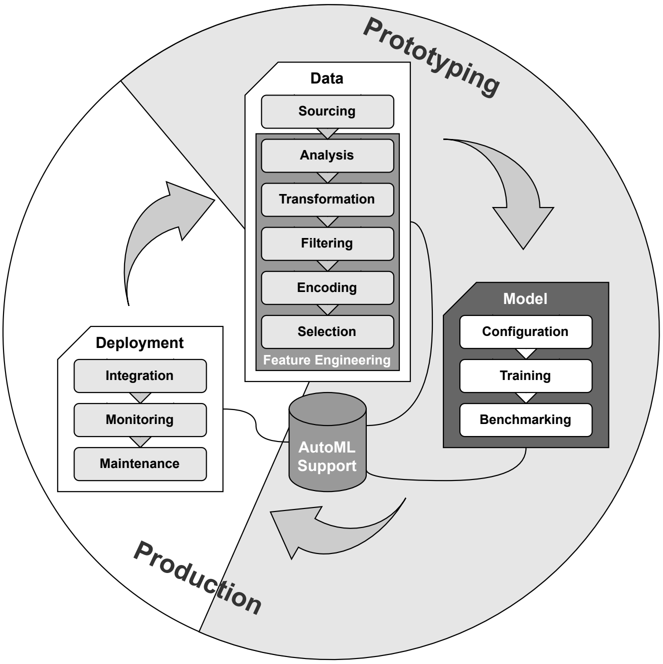

Figure 4.20: Block two of the method for AutoML-supported LTP: Model Development.

Configuration

The first step in this block is the *configuration*. This means setting up a pool of different pipeline combinations of ML algorithms and their corresponding hyperparameter settings. A model pipeline configuration consists of three items:

1. The *additional custom preprocessors* can be set to apply algorithm-specific alterations to the data which were not conducted in the data preparation block. Specifically, this refers to additional filtering or scaling methods and allows more fine-grained pipeline customisation as some ML algorithms require additional preprocessing steps in order to function properly. Thus, this item is entirely optional.
2. The *algorithm* refers to the concrete model type, i.e. RF, SVM, ANN, or any other kind of ML algorithm. As concluded in Section 3.1.4, studies have shown that especially tree-based algorithms, such as RF [PGKM16, GPN+18, LGA+18] or DT [Sau20, p. 191], often perform best in LTP. Other promising algorithms worth investigating include SVM [AMT08] and ANN [RW03].
3. The *hyperparameters* refer to the always algorithm-specific parameter-value-combinations. Finding the optimal hyperparameter values for any given data-algorithm-combination poses a search problem and requires a structured approach. Exploring as many viable combinations without bloating the search space is key. In online learning, this can be achieved by spanning a parameter grid with manually set value limits and step intervals (randomised or fixed) in order to maintain control about both the amount and variance of generated value combinations. This essentially leads to the initial situation as with the grid and random search approaches highlighted in Section 2.2.4 [BB12].

The above-mentioned approach of spanning a grid of viable hyperparameter value combinations to be explored is extended to the overall configuration step. In this, entire model pipeline configurations are created from various pre-selected preprocessors, algorithms, and hyperparameter value limits as well as step intervals. Based on this, the generation is automated by iterating through the grid, instantiating all model pipeline configurations within the defined pre-selections [MHM+21]. Figure 4.21 illustrates this idea.

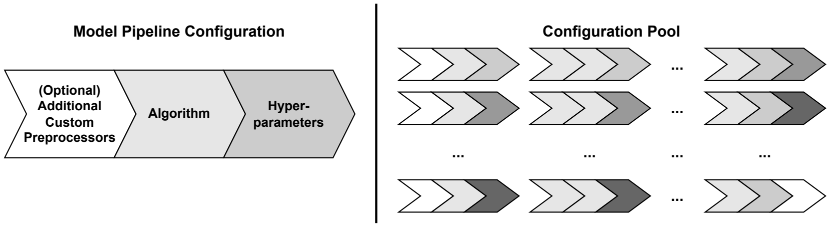
Model Pipeline Configuration
(Optional) Additional Custom Preprocessors
Algorithm
Hyper-parameters
Configuration Pool
...

Figure 4.21: Model pipeline configuration pool generation as based on [MHM+21].

This step is at the core of many AutoML solutions as was shown in Section 2.2.4, yet the techniques applied vary [HKV19, p. 3-33]. Unfortunately, especially many of the techniques for hyperparameter optimisation are not suited for online learning as not all data is available upfront. Therefore, the above-presented and easy to interpret generative approach is mandated here.

Training

Following the configuration of the search space, the *training* of the generated model pipelines commences. In an online learning scenario, this means that individual data points are fed to the model pipelines in parallel one-by-one in a vectorised fashion as soon as they emerge [HTF09, p. 397]. This mechanism can be adjusted for batch data by simply iterating over it, mimicking a data stream. Figure 4.22 visualises the principle.

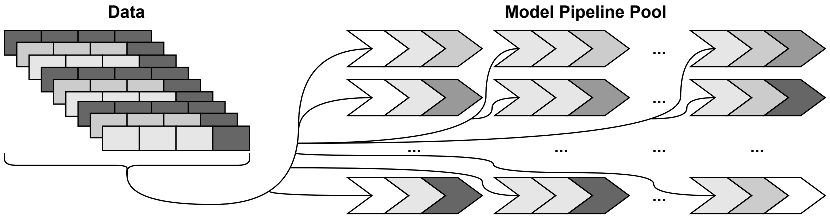

Figure 4.22: Iterative training of the model pipelines.

Training a pool of model pipelines in parallel is at the core of AutoML solutions [LP20]. However, depending on the size of the model pipeline pool, this endeavour can become quite resource-intense as expensive computations have to be performed for every model pipeline and both data and models have to be kept in memory. Therefore, the next section on the subsequent step discusses an approach to mitigate this overhead.

Benchmarking

The subsequent logical step is the *benchmarking* of the trained model pipelines in order to identify the best-performing model. First however, a suitable *metric* has to be chosen in order to compare different model pipelines according to how well they solve the problem of LTP. Table 4.3 summarises a selected set of metrics for both regression and classification performance assessments.

Table 4.3: Simplified summarisation of suggested suitable metrics for the benchmarking of LTP models. TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives, Truth = Ground Truths, Pred = Predictions

Regression	**Classification**
Mean Absolute Error (MAE) $$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert Truth_i - Pred_i\rvert \quad (4.3)$$	Accuracy (A) $$A = \frac{(TP_n + TN_n)}{(TP_n + TN_n + FP_n + FN_n)} \quad (4.4)$$
Mean Absolute Percentage Error (MAPE) $$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left\lvert\frac{Truth_i - Pred_i}{Truth_i}\right\rvert \quad (4.5)$$	F1 Score (F1) $$F1 = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \quad (4.6)$$
Mean Squared Error (MSE) $$MSE = \frac{1}{n}\sum_{i=1}^{n}(Truth_i - Pred_i)^2 \quad (4.7)$$	F1 Score - Precision $$Precision = \frac{TP_n}{(TP_n + FP_n)} \quad (4.8)$$
Root Mean Squared Error (RMSE) $$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Truth_i - Pred_i)^2} \quad (4.9)$$	F1 Score - Recall $$Recall = 2 * \frac{TP_n}{(TP_n + FN_n)} \quad (4.10)$$

Starting with the regression, four suitable metrics stand out. The simplest one is the MAE, as it can be intuitively interpreted as the model's mean deviation from the ground truth, given in the chosen time unit (e.g. minutes, hours, days). The MAPE relativises the MAE by outputting a mean percentage offset between the predictions and the ground truth. The MSE is again built on top of the MAE but by squaring the individual errors before summarisation and subsequent division, higher errors are granted more impact on the resulting score. Thus, the MSE reacts more sensitively when high individual errors are to be avoided in particular. However, the squaring potentially leads to results orders of magnitudes higher than the MAE, rendering it less interpretable. Therefore, the RMSE reduces it back to interpretable levels by applying an additional square root to the MSE. In case of higher errors during planning leading to disproportionally higher cost during production, the RMSE should be preferred over the MAE or MAPE.

In a classification, the simplest and most intuitive metric is the accuracy, whereby the amount of correct predictions is divided by the amount of all predictions, resulting in an easily interpretable percentage value. For a typically non-binary classification problem such as LTP, the accuracy is calculated on a per-class level and then averaged by weight in the class distribution. In cases with a low tolerance towards errors (false positives and false negatives), the F1 Score presents itself as an alternative. It is made up of the two components precision, i.e. the percentage of correctly identified positives out of all predicted positives, and recall, i.e. the percentage of correctly identified positives out of all true positives and false negatives. This grants more sensitivity towards misclassifications, especially false negatives. In the case that classification errors during the planning stage result in very high cost, the F1 score should be considered over the mere accuracy metric.

Following the selection of a suitable metric, the actual benchmarking of model pipeline configurations commences. Different techniques for this benchmark and selection process have been presented in Section 2.2.4. In order to provide a brief example, the method sequential halving is proposed here [KKS13]. In its simplest form, this algorithm consumes the population of model pipelines, the chosen metric, and a budget of cumulative model updates as inputs. Based on the size of the population and the set budget, it computes cut-off points, often called rungs, at which the worst-performing half of the population is discarded.

This process repeats until the budget is spent and the best-performing model pipeline configuration remains. Figure 4.23 provides an exemplary visualisation.

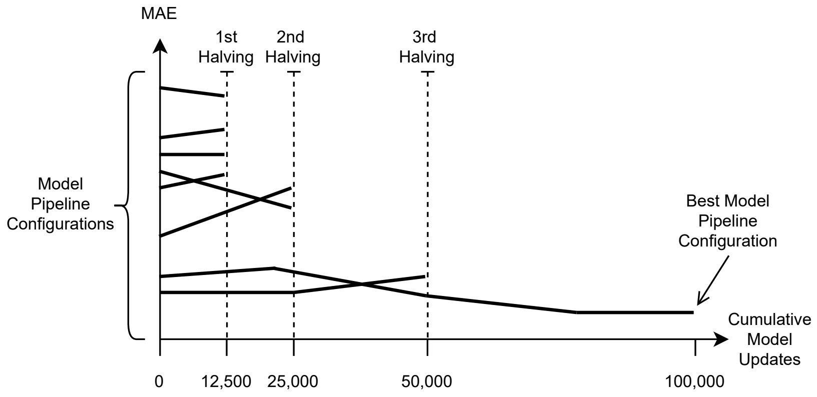

Figure 4.23: Example application of sequential halving minimising the MAE with eight model pipelines and a budget of 100,000 cumulative model updates. Based on [KKS13].

Sequential halving ensures that the most promising members of the population get to experience the most data while eliminating underperforming members early on in the process in order to preserve scarce computational resources. It is suitable for online learning as the data does not need to be available up front. Instead, it can run on a stream over time, learn incrementally, and eventually identify a well-performing model pipeline. Choosing a balanced budget however is a difficult task depending on the resources available and computation time decision makers are willing to allocate.

The benchmark step is very well covered by AutoML and similar solutions [MHM^{+}21]. However, making key decisions such as which metric to employ or how much computational budget to allocate for the sequential halving is still subject to expert judgement. Thus, experimentation during the prototype stage is strongly encouraged.

4.2.3 Model Deployment

Entering the third and final block *model deployment* marks the transition from the prototype to the production stage. In this, the LTP model is exposed to the real world in the three steps *integration*, *monitoring*, and *maintenance*. Figure 4.24 highlights the third method block.

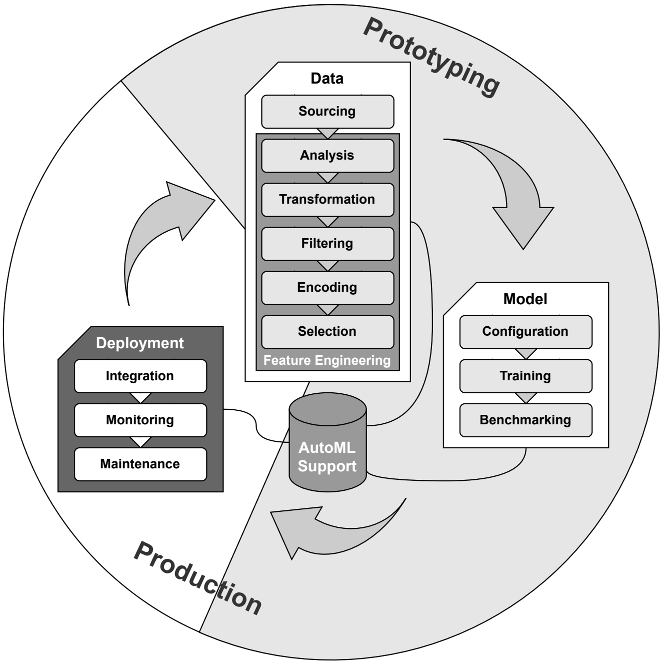

Figure 4.24: Block three of the method for AutoML-supported LTP: Model Deployment.

Integration

The first step of this block is the *integration* which refers to both the *technical* and *organisational* integration of the LTP model into a company's PPC. The technical integration follows a web-based containerised microservice approach [Wol17, p. 38-69]. It relies on REST in order to enable communication with a company's system landscape [Fie00]. Figure 4.25 provides an overview.

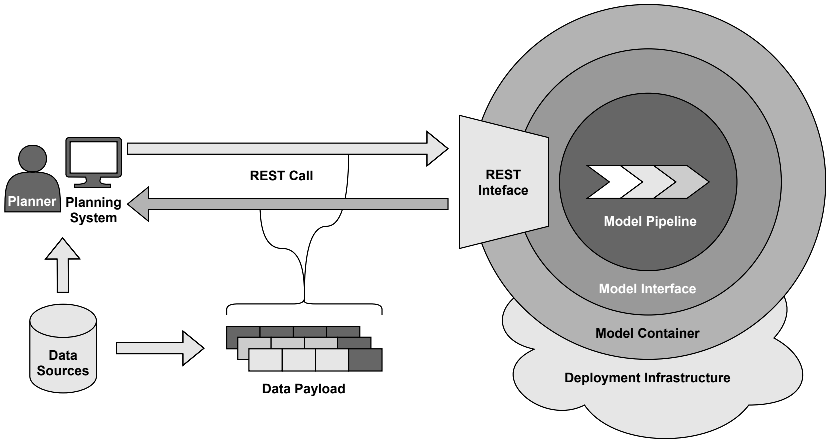

Figure 4.25: Technical integration of the LTP model.

The model pipeline is exposed through the model interface which is realised by the aforementioned REST. Using containerisation, the model container capsules the system environment in which the model pipeline's technical infrastructure is provided isolated from the actual system and hardware the container is running on. This in turn is provided by the deployment infrastructure which supplies the underlying computing resources and can be tailored to a company's abilities and requirements as needed. This way, the LTP model is capsuled independently from a company's system landscape and can be deployed anywhere within the network. Planning tools may interact with the system through its REST interface by providing the input data stream as new jobs arrive or plans need to be updated, while in return receiving the model's LT predictions.

Contrary to a direct integration into existing systems such as ERP or MES, the microservice approach generalises better towards a wide range of system landscapes present in SME as it is more flexible, scalable, and maintainable.

For the organisational integration, Sauermann et al. proposed a cascading closed loop model for PPC, which was discussed in Section 3.1.3 [SHPS19]. Aside from the conventional open PPC loop, this proposes a closed loop interlacing PPC including replanning activities with a prediction model control loop aimed at enabling model integration into PPC while at the same time maintaining sufficient model performance. This is expanded upon here as shown in Figure 4.26.

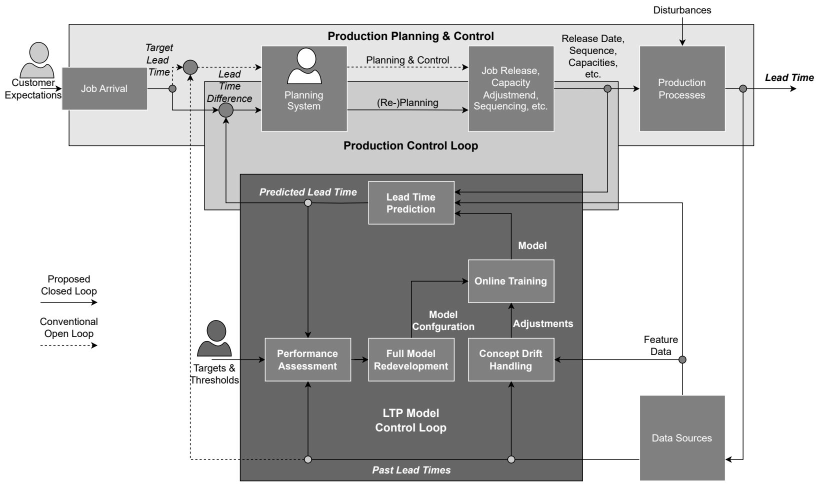

Figure 4.26: Organisational integration of the LTP model into PPC. Based on [SHPS19].

The LTP model control loop is altered to support online training as well as monitoring and maintenance. The last two come in the form of concept drift detection and adaptation, summarised under concept drift handling on the one side, and a full model redevelopment on the other side. Compared to the original concept by Sauermann et al., the concept drift handling allows to automatically cope with performance losses due to model degradation over time, at least within a reasonable limit.

In case the concept drift in the data is so severe that the model performance cannot not be retained by the concept drift handling, a full model redevelopment is triggered. These mechanisms are realised by the remaining method steps, detailed in the subsequent sections.

AutoML support for the integration step is limited. In terms of technical integration, some solutions offer low-level APIs which allow for functions such as model inference through web-based interfaces [LP20]. However, most solutions would require third party frameworks to realise the technical integration. The organisational integration is highly problem- and company-specific, therefore support by more generalised technical AutoML solutions is not available. Instead, inner-company PPC processes have to be adjusted on a case-by-case basis to leverage the full potential of the LTP model.

Monitoring

As previously stated, the next step *monitoring* is two-fold. The inner prediction model control loop deals with the concept drift detection (and subsequent adaptation). The outer loop entails conventional supervision of model performance indicators, such as the error metrics discussed in Section 4.2.2. Figure 4.27 zooms in on the LTP model control loop.

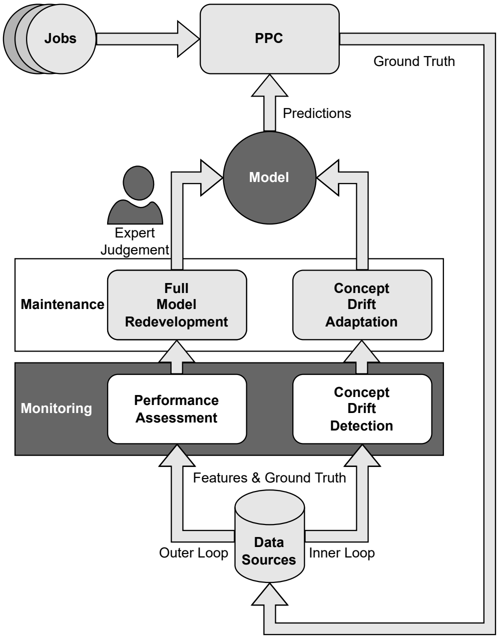

Figure 4.27: The monitoring step within the LTP model control loop.

For the inner loop, concept drift detectors monitor the input data stream and / or model-related performance criteria [GŽB+14]. Recommended are the in Section 2.2.3 aforementioned concept drift detectors EDDM [BGDF+06], ADWIN [BG07], or KSWIN [RHS20]. Typical parameters for these, such as window sizes and sensitivities, are problem-specific and need to be adjusted incrementally through experimentation. Some detectors explicitly express distinct stages. These are warning, when concept drift is likely to occur in the near future, and detection, when concept drift is actually detected.

Implementing the outer loop starts by settling for relevant performance indicators. Typically, this is the metric chosen in the benchmarking step. Other important metrics, depending on company-specific circumstances, could be the computing resource intensity or training and inference time constraints. For these metrics, *targets*, describing performance goals the LTP model should fulfil, as well as *thresholds*, meaning performance boundaries the LTP model must not undercut, need to be established by the management [SHPS19].

Maintenance

Following the monitoring approach, the last step *maintenance* is again conducted two-fold, along the inner and outer LTP model control loop. This is highlighted in Figure 4.28.

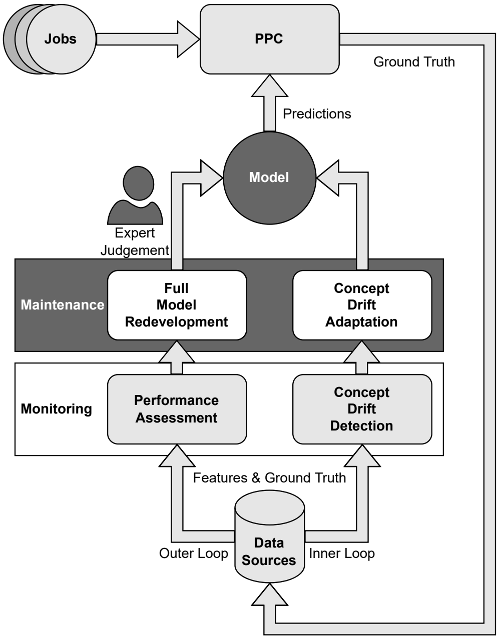

Figure 4.28: The maintenance step within the LTP model control loop.

On the inner loop, following a concept drift detection, the adaptation is triggered. Referring to Section 2.2.3, an informed adaptation strategy is chosen here. Depending on the model algorithm, this either results in a global or a local model replacement. In a global replacement, a new model with the same configuration is trained on the new data while in a local replacement, only parts of the existing model are replaced [GŽB+14]. This mechanism is implemented differently by various algorithms.

As an example for an algorithm providing local replacement capabilities, the *Adaptive Random Forest* (ARF) by Gomes et al. is highlighted here [GBR+17]. ARF is an extension of the RF algorithm which, amongst other features, provides online learning and concept drift handling capabilities. To do so, each tree within the forest commands its own drift detector, which is able to express both warnings and detections. Should a warning occur, a replacement tree starts to train in the background. On detection, this background tree is then switched with the active foreground tree, which is discarded afterwards. This mechanism ensures the ARF is robust towards various types of concept drift and can adapt efficiently. Pre-selecting algorithms such as the ARF in the configuration step earlier, ensures the inner LTP model control loop functions to its fullest capacity.

The outer loop serves as the re-entry point for the first method block. If performance degradation is so severe that the built-in concept drift handling mechanisms are unable to cope with the underlying changes in the data, a full model redevelopment is triggered. This means re-entering the data preparation block, especially focusing on the analysis block in order to identify the exact nature of the change but also to challenge previous assumptions.

AutoML support for both the monitoring and the maintenance steps is available but limited. It comes in the form of the aforementioned scheme of automated concept drift detection and handling, such as implemented in the ARF [GBR+17]. As maintenance is the last step, this concludes the main body of the method description. In the following, a suitable system architecture to implement the method is laid out.

4.3 System Architecture

Compared to the static scenario, operating within the streaming scenario requires a different architectural approach. In a static scenario, data, including the target variables' ground truths, is collected and then processed batch-wise to develop a predictive model, which is then deployed and can be inferenced at runtime. This means training and inference activities are strictly divided. The model cannot benefit from potentially new information contained in the data it inferences on at runtime. Of course, this data can be collected and used to trigger retraining at a later date, resulting in periodic model updates.

In contrast, in the streaming scenario both inference and training may occur simultaneously and iteratively. The online training algorithm learns from every example it is fed and builds up prediction performance over time. Although this happens at some delay. Naturally, at the time the prediction is required, i.e. during production planning, ground truth on the target variables is not yet available as production processes have not been executed and reported on yet. However, once ground truth does become available at a later date, it can be fed to the online model in order for it to learn from and improve itself. Instead of periodic model updates through retraining, online learning leads to frequent although smaller model updates. Figure 4.29 illustrates this idea.

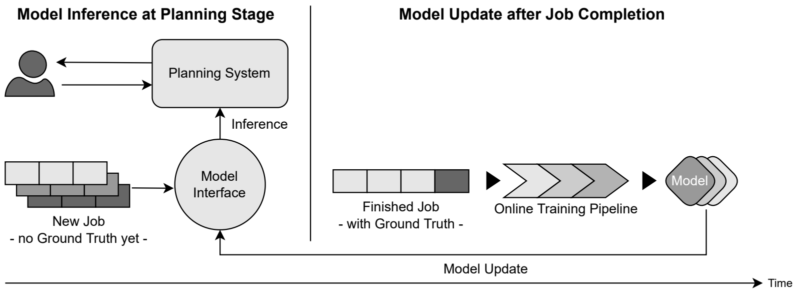

Figure 4.29: Basic principle of LTP online inference and training with delayed ground truth.

4.3.1 Inference Pipeline

To realise this approach, a different, streaming-oriented architecture is required. Figure 4.30 provides a sketch for the inference at the planning stage. As laid out in Section 4.2.1, data is entering the stream processing from different data sources. What then follows are implementations (processors) of the method steps transformation (including merge), filtering, encoding, and selection as detailed in that section. Data which went through this pipeline is then presented to the model interface for inference. The model's prediction is then passed on to downstream applications such as the planning system, with which the planner interacts.

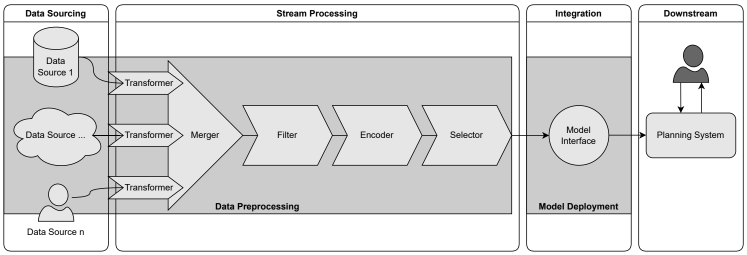

Figure 4.30: Architecture of the LTP streaming pipeline for inference.

4.3.2 Training Pipeline

The corresponding training pipeline is shown in Figure 4.31. The data flows through the same processors as in the inferencing case but then enters the model development block. In this, a concept drift detector will inspect defined variables, though at least the target variables, for concept drift. This detector follows a two-stage escalation scheme, defined by thresholds set in advance. The first stage is the warning stage in which anomalies have been detected but concept drift is not yet confirmed. The second stage is the actual confirmation of a detected concept drift.

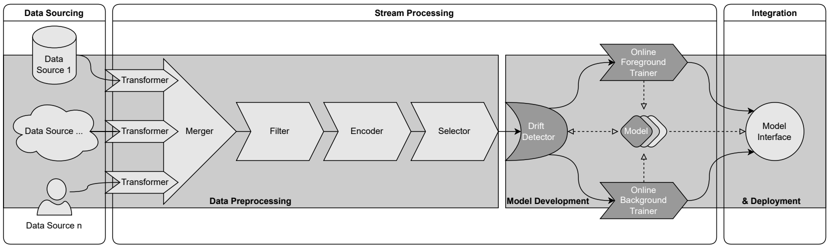

Figure 4.31: Architecture of the LTP streaming pipeline for training.

As presented in Section 4.2.3 and as long as no concept drift is present in the data, the foreground model or model part, i.e. the part that is currently deployed, is trained further. In case of a concept drift warning, a background model or model part starts to train. If the concept drift is confirmed at some point, the background model or model part is swapped in, replacing the foreground one. Finally, access to the model is provided via the model interface described in Section 4.2.3 which can be queried for inference. This concludes the main body of the method and system architecture description.

4.4 Assessment & Differentiation

To summarise this chapter, the thesis' main objective as defined in Section 1.2.1 was translated into three technical objectives in Section 4.1.1. These were then put into the context of a data streaming scenario with associated preconditions. Based on these, Section 4.2 then described a three block method for AutoML-supported LTP. The method was complemented by system architecture considerations in Section 4.3. In the following, both the conceptualised method and system architecture are assessed against the technical objectives as well as differentiated from the state of the art as it was laid out in Section 3.1. In addition, the degree of available AutoML support for the method is summarised as well.

4.4.1 Assessment of the Technical Objectives

To reiterate, the technical objectives of this thesis are defined in Section 4.1.1 as follows:

> TO1: Enable LTP by utilising AutoML to develop models capable of predicting PPR-specific PTs and TTs.
>
> TO2: Enable integration of the LTP models into a system landscape where they can be deployed, inferenced, monitored, and updated.
>
> TO3: Enable the LTP models to adjust to concept drifts in the input data in order to maintain a constant performance.

Assessing TO1, the first two method blocks data preparation and model development are specifically designed to fulfil this goal. While the first method block extracts and prepares data from PPR-specific data sources, the second method block produces models capable of predicting both PTs and TTs for single operations on given resources from this data. Both method blocks apply AutoML techniques where possible with a focus on model development activities.

TO2 is achieved in part by method block three with the integration, monitoring, and maintenance steps. In these, the deployment as a web-based containerised microservice, which can be queried and monitored via REST, is outlined. On a technical level, this is complemented by the training and inference pipeline system architectures laid out in Section 4.3. The update capability is a core mechanic of the online training approach laid out in the method steps training and maintenance. On an organisational level, an LTP model control loop fully integrated into PPC has been proposed in the integration step.

Lastly, TO3 is addressed via the method steps monitoring and maintenance. State of the art concept drift detectors are proposed for the identification of concept drifts in the input data. The adaptation is then realised by an informed strategy with a preference for local model replacements. However, the proposed LTP model control loop also allows for a full model redevelopment should local adaptations not have the desired effect. Thus, the technical objectives laid out in Section 4.1.1 are considered achieved by the proposed three-block method and the complementing system architecture.

4.4.2 Assessment of the AutoML Support

Table 4.4 summarises the degree of AutoML-support for each method step. Details on the individual step assessments are provided in the respective subsections of Section 4.2.

Table 4.4: Assessment on the degree of AutoML-support for each method step.

Block	Step	Degree of AutoML Support
Data Preparation	Sourcing	◔
	Analysis	○
	Transformation	◑
	Filtering	◔
	Encoding	◕
	Selection	◕
Model Development	Configuration	●
	Training	●
	Benchmarking	●
Model Deployment	Integration	◔
	Monitoring	◔
	Maintenance	◔

While the method block model development is adequately supported by AutoML, the method blocks data preparation and model deployment are supported in varying degrees only. Until more sophisticated AutoML solutions are available, especially data preparation remains the most laborious task of the proposed method.

4.4.3 Differentiation from Existing Methods for Lead Time Prediction

As shown in Section 3.1, a large body of work on LTP focused on the overall problem modelling, the application and comparison of individual ML algorithms as well as feature engineering techniques [RW03, ÖKÖ06, AMT08, dCGMT10, MLRH11, MDF+14, LYWF15, MM15, PGKM16, GPN+18, LGA+18, LYS19]. There was furthermore a schism between works focusing on PTP on the one hand and TTP on the other (see Tables 3.1 and 3.2 for references). In contrast to these earlier works, the holistic method presented in this thesis provides an end-to-end recipe ranging from data preparation and model development to the model deployment while at the same time unifying PTP and TTP into one approach. It also reduces the complexity of the overall task by employing AutoML techniques which is a novel approach in this problem domain.

Few authors, namely Pfeiffer et al. [PGKM16], Gyulai et al. [GPBG18], and Schuh et al. [SGS+20] acknowledged the problem of performance degradation through concept drift and proposed routine or performance-based model re-trainings. In opposition, the method presented here conceptualises a true online learning approach with sophisticated concept drift detection and adaptation mechanisms. Lastly, the integration into PPC as addressed in detail by Sauermann et al. was developed upon in this thesis in order to better incorporate said online learning approach [SHPS19, Sau20].

4.5 Chapter Summary

This Chapter described the main contribution of this thesis: the method for AutoML-supported LTP. It began by laying out the framework in which the method is meant to be operated. As such, the main objective from Section 1.2.1 was further systemised into technical objectives in Section 4.1.1. The data streaming scenario was outlined and preconditions for the method application were provided. This was followed by an in-depth description of the method which is organised in two stages, protoyping and production, dividing into the three blocks data preparation, model development, and model deployment. These three blocks are sub-divided into a total of twelve steps of which each provides concrete guidelines and makes suggestions on suitable methods for implementation. Subsequently, system architectures for inference and training pipelines based on the streaming scenario and the method steps were illustrated. Concluding the chapter, the designed method was successfully assessed against the technical objectives from Section 4.1.1. Furthermore, the AutoML support as provided by the state of the art was assessed based on its ability to cover each method step. This yielded mixed results, thus highlighting potential research gaps. Finally, the method presented in this thesis was sufficiently distinguished from the existing methods for LTP summarised in Section 3.1.

5 Validation

A Practical Application

In this chapter, the described method for AutoML-supported LTP is validated against two case studies. The first case study was already subject of a previous work [BTO22]. The main difference is that the original paper considered an offline learning approach whereas the method introduced in this thesis is based around online learning. The second case study is an alternative version of the prior, in which the data has been altered in order to assess the method's ability to handle concept drift. Details of both case studies are laid out in their respective sections below. In the final section of this chapter, based on the results of the case studies, the method is critically assessed.

5.1 Case Study A: Lead Time Prediction on Real-World Data

This first case study, referred to as case study A, revolves around performing LTP for an SME, mainly operating in the MTO domain. The case study was developed within the publicly funded research project *Alto* (Algorithm-based Optimisation of Timely Job Control in Make-To-Order Production), which ran from 2020 to 2022 [GBE+22]. Data produced by this case study was also used to conduct an LTP benchmark on AutoML solutions, although in a static, non-streaming context [BTO22].

5.1.1 Scenario

The company behind case study A is a Baden-Württemberg-based medium-sized enterprise mostly producing forging parts for various industries. The jobs are largely MTO with small series orders complementing the portfolio. The shop floor is organised as a flexible job shop, the smallest organisational units are machine groups of which there are 26. These realise a total of 22 different production processes. PPC is conducted manually, mainly optimising for adherence to schedule. The system landscape is defined by a proprietary ERP system which governs job data and shop floor reporting. There is a direct management access to the prioritisation of jobs via manually set priority flags. High priority jobs arriving on short notice frequently disturb operations. The shop floor operates in a single shift cycle on weekdays with occasional overtime working hours on Saturdays and seldom on Sundays. Reporting from the shop floor is handled manually via bar code scanning. During the project, digital production monitors had been installed at every work station, primarily displaying job priorities to the shop floor staff. Within the frame of the case study, the company provided 99,526 historical data points spanning the years 2019 to early 2021. These reference individual process executions and include expert estimations for the PTs, which served as a baseline performance indicator in the case study.

5.1.2 Method Application

Case study A covers the prototyping stage of the method with the two blocks data preparation and model development. The productionisation via the third block, model deployment, is not part of the case study. In the following, the application of the respective method steps is discussed. The prediction problem was modelled as a regression for both PT and TT with the MAE as target metric.

Sourcing

As stated before, 99,526 raw data points were sourced from the aforementioned ERP system. This data was outputted as CSV files and provided an already aggregated dataset. In order to mimic the streaming context, this data was later fed to the ML components incrementally, though without the delays that occurred in the real world. Table 5.1 summarises the key properties of the raw data. The entire column model, including brief explanations, is available in the appendix in Table A.1.

Table 5.1: Raw data overview.

General Properties	
Total Rows	99,526
Total Columns	30
From	08.01.2019
To	25.02.2021
Rows from 2019	42,625
Rows from 2020	52,156
Rows from 2021	4,745
Unique Values	
Jobs	15,478
Machine Groups	26
Processes	22
Materials	601
Target Variables	
Time Unit	Minutes
PT Baseline MAE	115.19
TT Baseline MAE	n/a

As stated before, the data contains expert estimations for the PTs. Stacked against the recorded actual PTs, i.e. the ground truth, a mean absolute error of 115.19 minutes was calculated. This serves as a performance baseline for the method results. Unfortunately, the raw data does not contain expert estimations on the TTs, thus there is no baseline available in this case. Concluding the sourcing step, Table 5.2 contains the considerations of the seven Vs as proposed in Section 4.2.1:

Table 5.2: Considerations of the seven Vs for case study A.

V	**Assessment**
Volume	In the data provided, there are 99,526 data points with 30 columns each, resulting in 2,985,780 values. Eight columns are deemed static master data. The ERP system already provides aggregated or computed values such as adherence to schedule.
Velocity	With the available data spanning 2.14 years and assuming a year has 250 working days, statistically roughly 186 data points are created per working day, or 23 per working hour. The creation rate does not follow a uniform distribution but instead fluctuates based on events occurring shop floor, seasonal trends, etc.
Variety	As the data is already aggregated by the ERP system, it arrives in a structured form. However, data types vary between numeric, string, date, and datetime. Strings are often aggregated, representing multiple pieces of information.
Veracity	Parts of the data are managed manually. This relates to static master data, such as materials and their properties but also to the PTs and TTs recorded by barcode scanning. These manual inputs reduce the overall trust in the data as errors have to be expected.

Value	A value assessment on different data sources cannot be produced here as the ERP system is the only source available. However, the columns contain some redundancies which can be removed safely without reducing the data's overall value. The data expresses information on products, processes, resources, and the target variables, which weighs in its favour. However, the information expressed is superficial and lacks detail.
Variability	A variability assessment on different data sources cannot be produced here as the ERP system is the only source available.
Volatility	Given the data stretches over the beginning of the COVID-19 pandemic in 2020, it is plausible that concept drift occurred due to the pandemic's effects on public health and changes in the German legislation to contain the pandemic as well as subsequent effects on the company's processes and staff availability. However, a clear indication can only be given after completing the first iteration of the method cycle.

Analysis

Analysis of the raw data showed an amount of NaN-values which are quantified per column in the appendix in Table A.2. These are exclusively due to missing values. At first glance, corrupt or unintelligible data was not identified. Six categorical columns were determined, which are displayed in the appendix in Table A.3. The most important statistical properties were calculated for the target variables' columns, as shown in Table 5.3.

Table 5.3: Statistical properties of the target variables in the raw data.

Statistic	**PT**	**TT**
Mean	130.19	1,945.39
Standard Deviation	331.31	3,703
Min	0.00	0.00
25%	8.92	0.00
50%	33.23	270.21
75%	120.07	2,485.00
Max	20,828.28	161,299.82
Share of Total LTs	6.27%	93.73%

As expected, PTs are in general much lower than TTs. Interesting to note is the large spread between minima and maxima on both target variables. Especially the occurrence of zero values, but also the extremely large maxima values in the thousands, indicate the presence of outliers in the data. This is even the case despite the data being corrected with regards to non-working hours by the ERP system beforehand. About 14% of the PTs and 33% of the TTs in the raw data are between zero and one minute. These values seem implausible though could be explained by human errors in the manual barcode scanning. I.e. staff failing to report process executions entirely or reporting them in bulk. It was not possible to establish the precise reason for these outliers during analysis. The high range also impedes the suggested visualisation of the target variables' distributions as histograms and box plots. For better visibility, Figure 5.1 shows scatter plots of the target variables plotted against the timeline.

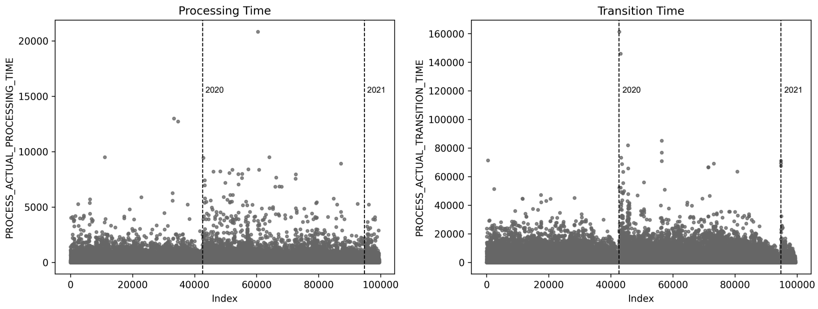

Figure 5.1: Target variables in the raw data over time.

In this, the extremely large outliers become clearly visible with the overwhelming majority of the data points accumulating at the bottom of both plots. There are some similarities for both target variables with plunges around the 40,000 and 95,000 data point marks and subsequent rises shortly after. These can be explained by the Christmas and new year's holidays taking place around the time frames these data points fall into. A visible effect of the COVID-19 pandemic cannot be corroborated here. Further analysis into the relationships between the target variables as well as the baseline is shown in Figure 5.2.

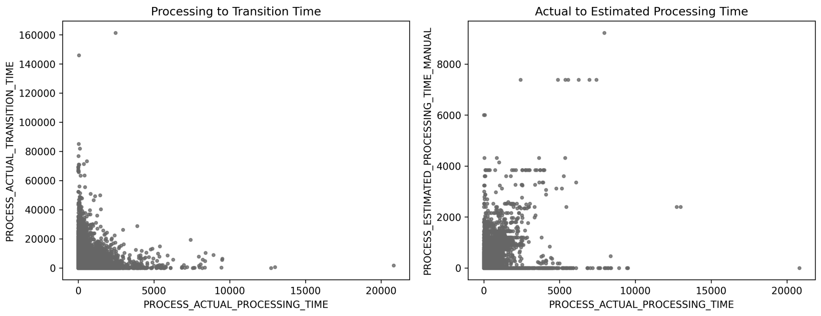

Figure 5.2: Both target variables in relation (left) and the actual PT in relation to the expert estimation (right) in the raw data.

As appears on the plot, the target variables behave slightly disproportional to one another. However, correlation analysis shows a very weak positive coefficient of 0.07 between the target variables. The baseline estimation for the PT appears to underestimate the ground truth but correlation analysis shows a weak positive coefficient of 0.35. Further correlation analysis on the numerical columns yielded no strong correlations. This is apart from the fact that processes and machine groups naturally form pairs as the latter are typically clustered by the processes their support. In the raw data, 59 unique process-machine-group-combinations were identified. This means there is an n:m relationship between processes and machine groups which execute them. Continuing, Figure 5.3 informs about the most commonly executed processes derived from the raw data.

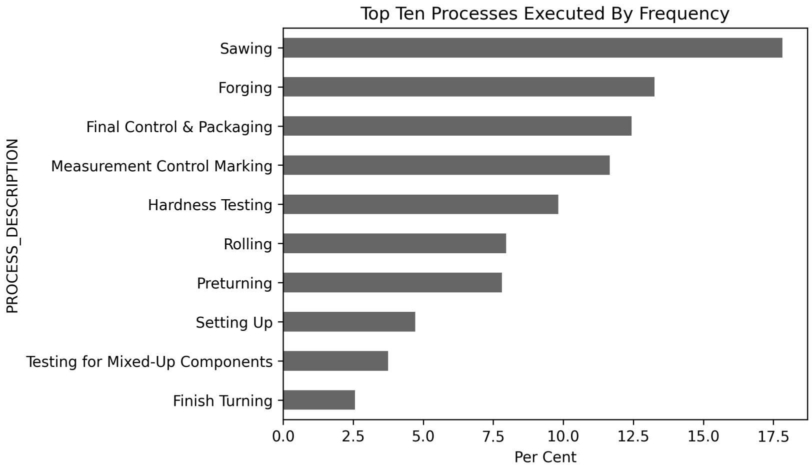

Figure 5.3: Top ten processes executed in the raw data by frequency.

These top ten processes account for a combined 91.81% of all processes executed within the raw data. Sawing and forging typically provide the first two steps in any job sequence. From there, these sequences branch into more product-specific processes. When analysing the share of PT per process over all accumulated PTs, the list shifts slightly.

Figure 5.4 shows that preturning makes up more than a quarter of all PTs with sawing being second just under 25%. PT-wise, forging only makes up only around 4% of all accumulated PTs.

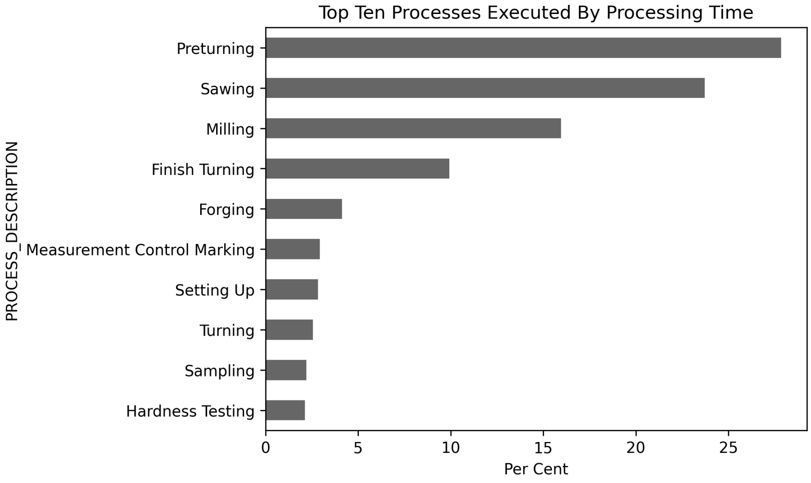

Figure 5.4: Top ten processes executed in the raw data by their share of the total PTs.

It is worth pointing out that the raw data contains setup processes as well as maintenance or holding processes which are not directly furthering the product. Also, the company distinguishes the stages of a turning activity into different processes. The complete list of processes and their shares within the raw data is given in the appendix in Tables A.4 and A.5. In addition, the PTs and TTs per process over time are provided in the appendix in Figures A.1 and A.2 respectively.

Following the individual process analysis, more insight was gained by reviewing the process chains which are realising the job executions. As there are 15,478 jobs in the raw data spread over 99,526 data points representing individual process executions, statistically each job is realised by a process chain with a length of six. However, it is worth noting that some jobs only appear with one or two process executions in the raw data.

In some cases, this is due to the fact that jobs started or finished outside of the 2019 to 2021 data window and are thus only partially available. In opposition, there are some jobs with multiple rework processes, leading to longer process chains. The longest process chain on record is 47 long. Figure 5.5 visualises the most often occurring process chains, capping the visualisation at six.

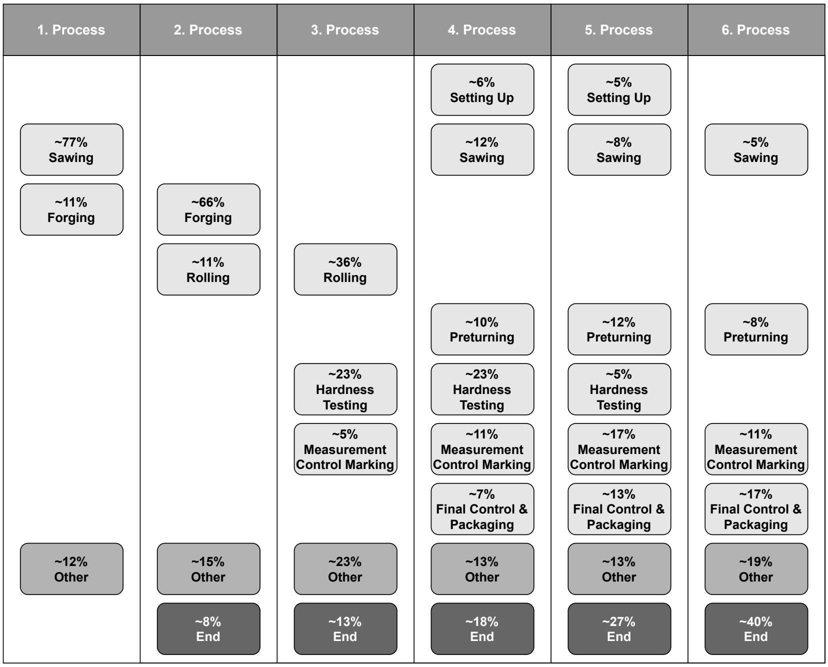

Figure 5.5: Process chains as occurring in the raw data. Processes with a link share below 5% are summarised under Other. Only the first six links are displayed.

A typical job starts with a sawing process followed by forging, rolling, hardness testing, and ends with final control and packaging. However, for the 15,478 jobs in the raw data, 4,690 uniquely different process chains were identified. And this does not include varying parameters within these processes themselves. This goes to highlight the great variety in MTO production where every job is differing from the next one in some aspect.

Continuing the analysis, further insight was gained by visualising the processed materials, as shown in Figure 5.6.

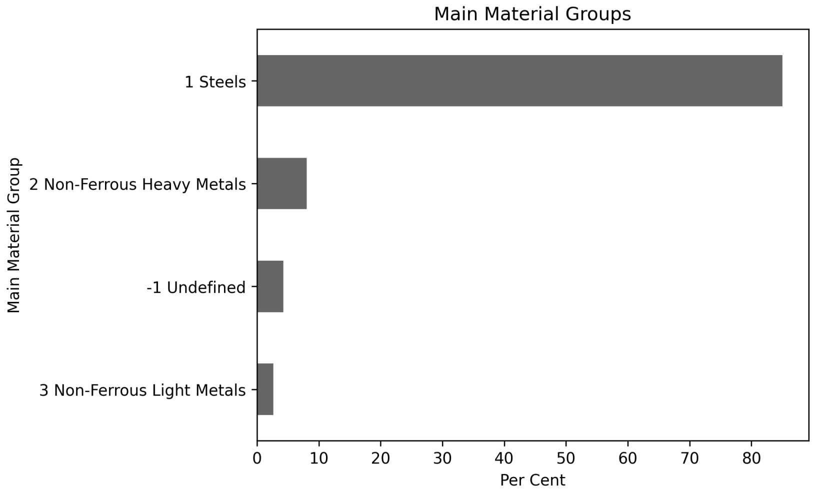

Figure 5.6: Main material groups according to DIN EN 10027-2:2015-07 [DIN15] in the raw data by frequency.

The classification is based on the DIN EN 10027-2:2015-07 [DIN15], which dictates a scheme of main material groups and associated subgroups. Displayed here are only the main material groups. As derived from the raw data, out of the 601 different materials processed by the company 85% belong to the group of steels, followed by the group of non-ferrous heavy metals at 8%. About 4% cannot be identified by this scheme as the material number is incomplete or missing from the data. The remainder is made up of non-ferrous light metals.

Further analysis performed on the remaining columns lead to the detection of sporadic anomalies where date columns contained implausible values. For example, some job release dates were earlier than job approval dates. These were attributed to system defaults automatically filling NULL values.

In summary, the analysis yielded the following key insights:

- Both PTs and TTs show high variances and outliers in the form of implausibly low and high values, indicating an overall poor data quality. The reasons are unclear but it is likely that human errors in the manual barcode scanning before and after each process execution are one major cause. Thus, these outliers need to be addressed in the subsequent steps.
- PTs only make up 6.27% of the total LTs, falling in line with previous findings from other studies [SPSF19, SPM+19, SGST20]. However, this assertion is under limited confidence due to outliers present in the data.
- Both PTs and TTs had already been corrected for non-working hours by the ERP system.
- The top ten processes by frequency account for over 91.81% of all processes executed. If measured by PT share, this number goes up to 94.25%.
- A typical job is realised by six different processes. Though the data shows a high variance with the amount of processes per job ranging from one to 47.
- In 85% of the jobs, a type of steel (main material group 1 as per [DIN15]) is the main material.

Transformation

During transformation, six major adjustments were performed on the data. These are summarised in Figure 5.7 and laid out in detail below.

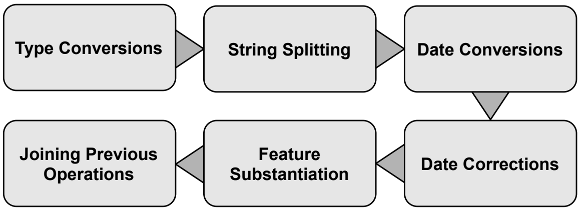

Figure 5.7: Transformations applied to the raw data.

First, type conversions were performed, where possible from string to numeric values. Within this adjustment, the NaN values were replaced by sensible defaults. This was followed by the string splitting, where strings were broken up into different columns, often followed by another type conversion, to separate the information within them. In some instances, characters were dropped entirely. An example is breaking up the six-character material number into the main material group (1st character), sub material group (3rd and 4th characters), and counter (5th and 6th characters). In this instance, the separator (2nd character) was dropped, as it represented no useful information. In the next adjustment, all date and datetime columns were converted into Unix notation as numeric values in milliseconds. Subsequently, the date anomalies detected during the analysis step where corrected by adjusting job approval and release dates to the same day. In the following, results of the string splitting were used to perform feature substantiation by creating new features from the individual split components. Further substantiation was achieved by recombining different features. I.e. distances between absolute date values have been calculated to create features representing time spaces instead of points in time. In the last adjustment, for each row, the rows of the previous operations were joined to it. This ensures a learning algorithm can harness information on previous operations as was suggested by Meidan et al. [MLRH11]. The adjustment was performed based on the previous operation within the same job but also based on the previous operation on the same machine group.

This last adjustment of joining of previous operations has two caveats which need to be highlighted here. For once, it varies how many machines make up one machine group. A portion of the machine groups in the company are effectively only one machine or work station but others operate multiple machines in parallel. Thus, the previous operation might have been executed on a different machine and has no impact on the job at hand. In order to maintain a consistent column model, this was not distinguished further and the caveat was accepted. The more pressing concern with this approach though is the fact that, depending on the time span between the conclusion of the previous operation and (re-)planning the current one, this information needs to be considered a posteriori. It thus would not necessarily be available at the time of prediction. For purposes of investigating the potential impact of information on previous operations, this adjustment was done regardless. The resulting column model is available in the appendix in Table A.6.

Filtering

The filtering of the transformed data was conducted separately to create filtered data tailored to both target variables. Table 5.4 provides an overview of the filters and the number of removed rows.

Table 5.4: Filters and their effects on the transformed data.

Filter	**Rows Removed**	**Rows Remaining**
Processing Time		*99,526*
IQR Factor 1.5	11,507	88,019
Material Type Undefined	3,590	84,429
Processing Time <= 1	13,480	70,949
Irrelevant Process	4,674	66,275
Transition Time		*99,526*
IQR Factor 1.5	9,307	90,219
Material Type Undefined	3,816	86,403
Irrelevant Process	4,119	82,284

The IQR filter refers to the interquartile range method introduced in Section 4.2.1. Its purpose is to remove the outliers and it was only applied to the respective target variable with a factor of 1.5. The next filter removed all data points with undefined materials. For the filtered data for the PT prediction, another filter was applied which removed any value up to one minute to root out implausible and impossible PTs. This was not performed for the TT variant as additional analysis revealed some consecutive operations occur within the same machine group, rendering it likely that at least a portion of zero or near-zero TTs are plausible and genuine. The last filter takes out rows containing processes deemed irrelevant. These extend to setup, waiting, maintenance, and downtime process dummies. Plots of the resulting filtered data are available in the appendix in Figures A.3 and A.4.

Encoding & Selection

The last two steps of the data preparation are combined here as they were implemented in their simplest form. For the encoding, a label encoder was used to encode 15 categorical columns. This includes columns added during the transformation step. The selection was performed manually by removing job IDs, dates (points in time), manual estimations for LTs and PTs as well as the a posteriori columns, including the ground truth which was set apart for the training step. The resulting column model spans 60 columns and is available in the appendix in Table A.7. This concluded the first method block data preparation.

Configuration, Training, & Benchmarking

Beginning the second method block model development, considerations went into the pre-selection of learning algorithms and their hyperparameter boundaries for the configuration step. As was shown in Section 3.1.4, tree-based methods performed well on LTP [PGKM16, GPN+18, LGA+18]. Therefore, online learning counterparts of the classic RF were pre-selected. This choice fell on the aforementioned ARFR by Gomes et al. [GBR+17] and the *online extra trees regressor* (OXTR) by Mastelini et al. [MNVdC22]. A parameter grid was laid out for both algorithms. Boundaries for the most important parameters are provided in the appendix in Table A.8.

Feature scaling was applied as custom preprocessor. Technically, this should not be important with tree-based algorithms but preliminary experimentation showed a positive effect on performance. Hence, a total of 540 model pipelines were generated for each target variable.

The in Section 2.2.4 proposed sequential halving algorithm by Karnin et al. [KKS13] was then applied to the configuration pool in order to identify the most promising model pipelines based on the MAE metric. This was done once for each target variable, PT and TT. Therefore, a budget of 50,000 cumulative model updates per run was set. The entire data was fed to the algorithm incrementally in order to mimic the online learning scenario.

As stated before, the baseline to validate against were the expert estimations contained in the data. These however were only available for PTs. In order to establish a secondary baseline for a data-driven model for both target variables, a lightweight statistical approach, the *rolling mean model* (RMM), was implemented as well. This was a simple table structure of rolling mean values which were continuously computed for each process. As predictions, the current rolling mean values for the processes in question were returned.

5.1.3 Results

In order to analyse the results, the selected hyperparameters and performance-related outputs were tracked. Table 5.5 summarises the results of the ML models produced by the method in opposition to the aforementioned RMMs and the expert estimation baseline.

Table 5.5: Best-performing model pipelines versus the RMMs and baseline.

(Hyper-)Parameter	**Processing Time**	**Transition Time**
Best ML Model		
Algorithm	ARFR	OXTR
n_models	50	100
max_features	25%	25%
max_depth	None	5
min_samples_split	3	3
MAE	35.95	880.15
Improvement over Baseline	37.96%	n/a
Improvement over RMM	03.41%	05.74%
Rolling Mean Model (RMM)		
MAE	37.22	933.73
Improvement over Baseline	35.77%	n/a
Baseline		
MAE	57.95	n/a

The ML models produced by the method performed best for both target variables, however the RMMs come in as close seconds. Far off for the target variable PT resides the expert estimation baseline. Figure 5.8 visualises the performance over time for both target variables. On the x-axis lies the data point index, the y-axis marks the MAE.

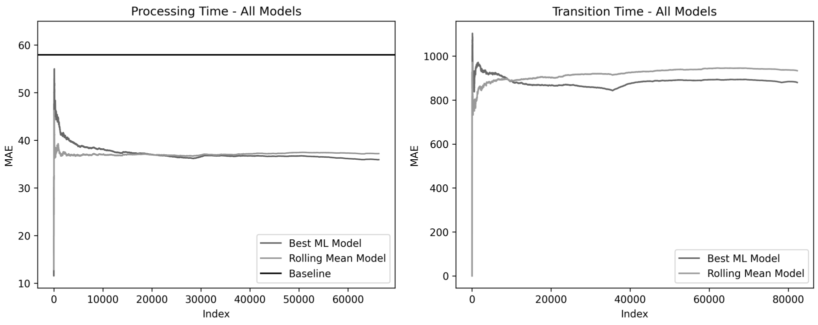

Figure 5.8: Performance over time of the best ML models against the RMMs and the baseline.

The expert estimation baseline is drawn as a constant. The curves for the data-driven models show that the rolling mean models start off with a lower error rate but are undercut by the ML models at some point. Furthermore, for PT, the slight downward slope of the error curve indicates a potential for a performance increase if more data was available. However, with the currently available data, the ML models only perform slightly, though consistently, better than the RMMs. Noteworthy for all models is a consistent rise of the error rate around the 30,000th data points index which remained unexplained.

Error Analysis

The performance represented by the MAE was analysed further to gain deeper insights. Figure 5.9 shows the MAE on a per process level for the PT prediction.

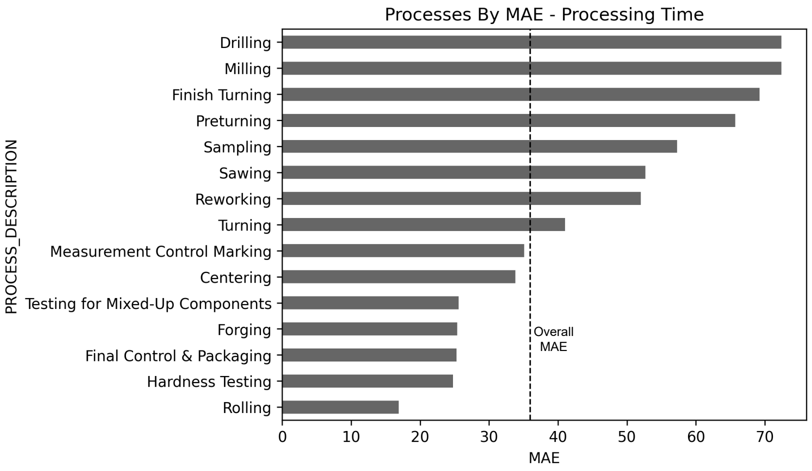

Figure 5.9: MAE of the PT ML model for each process.

As seen by the above average error rates, drilling, milling, (pre- / finish) turning, sawing, and reworking processes are harder to predict for the ML model than the other processes. However, this illustration does not take into account the frequency of the processes, i.e. how commonly these are executed. As was shown in Figure 5.3, sawing makes up about a fifth of all processes executed. Preturning accounts for less than 10%, finish turning for around 2.5%, and the other hard to predict processes for even less. Therefore, a more useful conclusion can be drawn if the process-individual error rates are weighted by their frequency in the data. Figure 5.10 illustrates this adjustment.

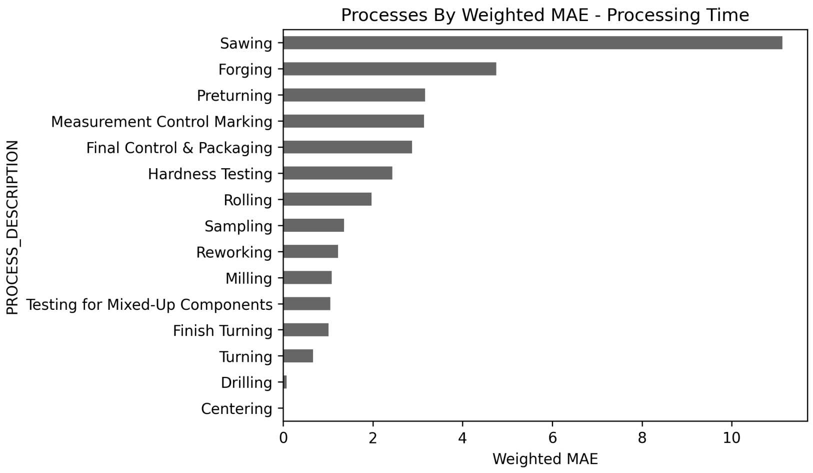

Figure 5.10: MAE of the PT ML model weighted by the frequency share of each process.

When adjusted for frequency, the sawing process stands out as both often executed and hard to estimate for the ML model. This is followed by the forging process which however shows a below-average MAE and preturning which, similar to sawing, occurs frequently but is difficult to predict. Discussions with the experts at the company revealed that the sawing process is problematic in various ways. The machine group in question was understaffed and underequipped yet came into play early on in most jobs. Hence, delays or inaccurate estimations affected the entire schedule, negatively impacting the overall adherence. This bottleneck was later addressed by installing multiple new sawing workstations.

The same analysis was conducted for the TT prediction. In Figure 5.11, the MAE on a per process level is depicted. Only five processes show a below average MAE.

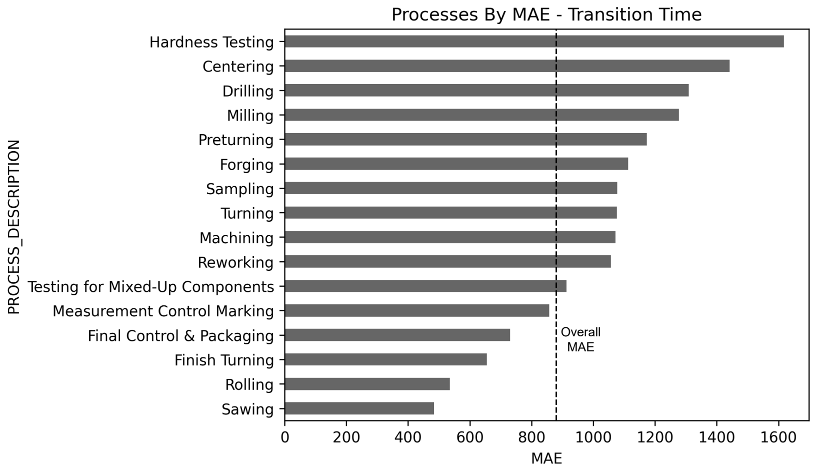

Figure 5.11: MAE of the TT ML model for each process.

Applying the weighting approach introduced before, Figure 5.12 shows the MAE adjusted for frequency. This changes the picture in that both forging and hardness testing occur frequently but are hard to predict TTs for. The fact that forging and the subsequent hardness testing are typical direct follow-ups to the already problematic sawing process, presents a possible explanation for this occurrence.

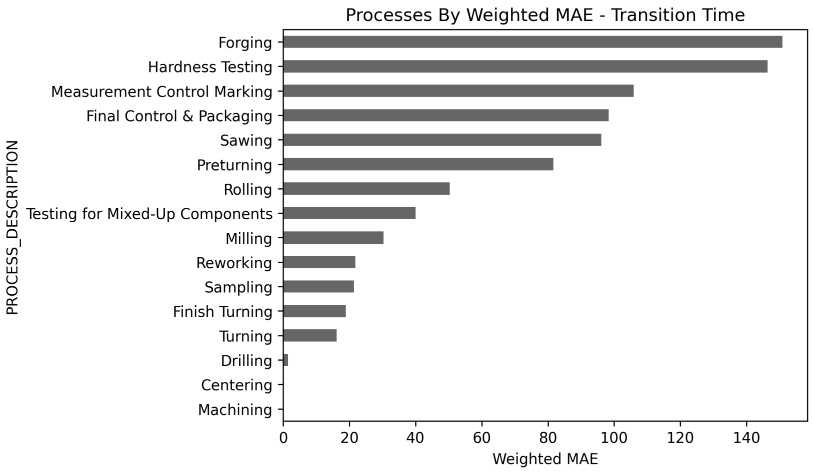

Figure 5.12: MAE of the TT ML model weighted by the frequency share of each process.

Feature Analysis

Complementing the error analysis, further investigation on the predictive capabilities of the selected features was performed. In order to calculate these feature importances, a permutation feature importance approach was applied [Bre01]. In this, consecutive testing runs are performed in which individual feature values are permuted, thus breaking the relationship between the features and the target variable. On each run, the model performance is measured in order to determine the impact of the permuted feature, returning a feature importance score. This is only an approximation of the predictive capability though as it only measures the importance an individual feature has to a concrete model. Drawing a generalised conclusion on the predictive capability of an individual feature is not possible with this approach. Furthermore, a testing set of data previously unseen by the model is required, hence the trained online model cannot be reused for this analysis. Therefore, offline proxy counterparts to the best performing online models were trained using the same hyperparameters and an 80%-20% train-test-split. Table 5.6 shows the results for the PT.

Table 5.6: Top ten permutation feature importances of the PT proxy model.

Feature	**Score**
PROCESS_ID	0.240 ± 0.004
MACHINE_GROUP_ID	0.194 ± 0.003
PREV_PROCESS_ID_MACHINE_GROUP	0.098 ± 0.002
DATE_PLANNED_DELIVERY_ INTERNAL_EXTERNAL_DELTA	0.029 ± 0.001
PREV_MACHINE_GROUP_ID_JOB	0.019 ± 0.001
CUSTOMER_ID	0.017 ± 0.001
PART_DESCRIPTION	0.016 ± 0.001
PREV_PROCESS_ID_JOB	0.016 ± 0.001
PREV_MACHINE_GROUP_LOAD_ MACHINE_GROUP	0.015 ± 0.001
MATERIAL_SUB_TYPE_INDEX	0.011 ± 0.001

The highest importance to the model has the PROCESS_ID, which is expected considering the comparatively good performance of the RMM, which itself is centred around this feature. Next up is the MACHINE_GROUP_ID which, as the analysis method step showed, is correlated to the PROCESS_ID in that each process can only be executed on a specific machine group or set of machine groups. Although there is an n:m relationship, these two features have to be considered weak proxies for one another. Lastly, looking at the third highest importance, is interesting to note that the process previously run on the same machine group carries some importance. Below these however, the feature importance scores drop drastically. The same analysis was performed for the TT prediction with the results shown in Table 5.7.

Table 5.7: Top ten permutation feature importances of the TT proxy model.

Feature	**Score**
PREV_MACHINE_GROUP_ID_JOB	0.087 ± 0.001
PREV_PROCESS_ID_JOB	0.059 ± 0.001
PREV_DATE_PLANNED_DELIVERY_ INTERNAL_EXTERNAL_DELTA	0.057 ± 0.001
PROCESS_ID	0.054 ± 0.001
PREV_MACHINE_GROUP_ ATTENDANCE_RATE_JOB	0.051 ± 0.001
MACHINE_GROUP_ID	0.040 ± 0.001
PREV_RUN_ID_JOB	0.018 ± 0.000
PREV_WORK_SEQUENCE_ID_JOB	0.015 ± 0.000
WORK_SEQUENCE_ID	0.009 ± 0.000
RUN_ID	0.008 ± 0.000

As expected, features describing relationships to previous processes or machine groups affect the TT prediction the most. A familiar pattern emerges as features denoting processes and machine groups score highest. Features relating to the production sequence also play a role, though it is a minor one. However, feature importance is generally on a lower order of magnitude than in case of the PT prediction.

5.1.4 Discussion

In summary, case study A applied the first two blocks of the method for AutoML-supported LTP as described in Section 4.2 to a real world dataset from a medium-sized company of the MTO domain. When pitted against expert estimations and a simple RMM, it was determined that the method consistently produced the highest-performing predictive models. It was found that the error rates showed high variances across the different processes. For some processes, estimating LT was harder than for others. Furthermore, the performance gap between the ML models and their rolling mean counterparts leaves room for improvement. As was analysed, one explanation for this is the overall poor data quality of the real-world dataset. Thus, the best course of action in order to leverage the full potential of the method is to improve the quality of the input data.

5.2 Case Study B: Lead Time Prediction under Concept Drift on Altered Real-World Data

The second case study, referred to as case study B, is built upon case study A. It introduces artificial concept drift and is designed to assess the capability of the method to mitigate this negative impact on the model performance.

5.2.1 Scenario

The basic scenario is the same as in case study A. However, since it is unknown whether concept drift is present in the original data, it was altered by deliberately introducing artificial concept drift. Based on the concept drift patterns introduced in Section 2.2.3, the four sub-scenarios *sudden*, *incremental*, *gradual*, and *reoccurring* were created [GŽB+14]. This was conducted for both target variables. In order to produce comparable results, several assumptions were established across all sub-scenarios:

- Filtered data from case study A was used as a basis in order to remove the impact of outliers on the target variable distributions.
- Concept drift only occurred on the target variables.
- Only one pattern of concept drift occurred per sub-scenario.
- Concept drift only affected the latter half of the data, except for the reoccurring drift sub-scenario as this normalises in between cycles.
- At maximum, 75% of the data points within the affected portion of the data were altered.
- Concept drift magnitude was set to a maximum of two standard deviations of the respective target variable. Only positive concept drift, i.e. an increase of the respective target variable's value, was applied.

Figure 5.13 illustrates the resulting four concept drift data sets for the target variable PT in comparison to the original data. The resulting image for the TT looks similar and is therefore omitted here though attached in the appendix in Figure A.5.

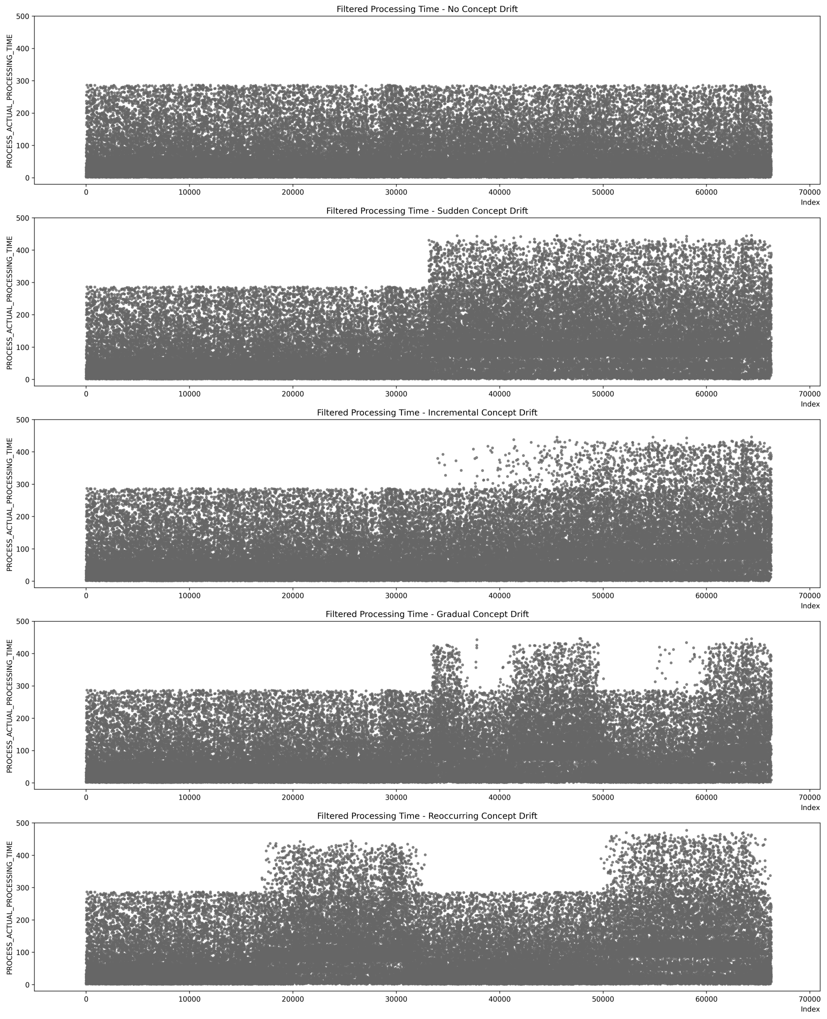

Figure 5.13: The original filtered processing time in comparison to the altered datasets of all four concept drift patterns.

As intended, the sudden drift sub-scenario shows an abrupt shift on the latter half of the data, immediately plateauing out on the new normal. The incremental drift sub-scenario is similar to the sudden drift one but instead, the shift occurs steadily over time. In the gradual drift sub-scenario, the shift is bouncing up and down before plateauing out. Finally, the reoccurring drift sub-scenario establishes a repeated upward and downward cycle which does not plateau out. All of these concept drifts are deliberately breaking the relationships between the features and the target variables, thus forcing the models to adapt to the changing circumstances in order to maintain performance.

5.2.2 Method Application

The method was applied for each of the four sub-scenarios separately. As the data was already transformed and filtered, the method application started at the encoding step and then proceeded exactly as in case study A. The third method block model deployment was partially realised in order to examine how the concept drift detection and adaptation mechanisms would function when confronted with this altered real-world data. Therefore, the inner loop of the monitoring step was implemented by applying ADWIN concept drift detectors with a warning-detection scheme as laid out in Section 4.2.3 [BG07]. As for the concept drift adaptation in the subsequent maintenance step, an informed strategy with local replacement was chosen. This was realised by the aforementioned ARFR [GBR^{+}17] and OXTR [MNVdC22] algorithms. Lastly, the RMM was trained as a baseline in the same way as it was done in case study A.

5.2.3 Results

The selected hyperparameters and performance outputs for all four sub-scenarios were tracked in the same way as in case study A. For a more concise reporting in the following, the sub-scenarios are subsumed under the target variables.

Processing Time

Table 5.8 summarises the results for the target variable PT in all four sub-scenarios. The best ML models resulting from the method application are compared to the RMMs and the unchanged expert baseline estimations.

Table 5.8: Best-performing model pipelines versus the RMMs and baselines for the PTP under the concept drift scenarios sudden, incremental (inc.), gradual, and reoccurring (reoc.).

(Hyper-)Parameter	**Sudden**	**Inc.**	**Gradual**	**Reoc.**
Best ML Models				
Algorithm	ARFR			
n_models	75			
max_features	25%			
max_depth	None			
min_samples_split	5			
MAE	44.12	43.85	42.81	46.71
Improvement over Baseline	50.00%	37.59%	41.58%	44.13%
Improvement over RMM	14.91%	05.84%	09.89%	13.90%
Rolling Mean Models (RMMs)				
MAE	51.85	46.57	47.51	54.25
Improvement over Baseline	41.24%	36.43%	35.17%	35.12%
Baselines				
MAE	88.24	73.26	73.28	83.61

The best ML model pipelines share the same algorithm and hyperparameters across all four sub-scenarios. The recorded MAEs show a drastic improvement ranging from 37% to 50% when compared to the unadapted expert estimation baselines, which themselves are surging due to the underlying concept drifts. However, the ML models also show an improvement between 5% and 15% in all four sub-scenarios over the RMMs.

Especially in the sudden and reoccurring concept drift sub-scenarios, the RMMs are unable to adapt quickly to the changing distribution of the target variable and thus get outperformed by the ML models. Figure 5.14 illustrates the performance over time of all models in every of the four sub-scenarios.

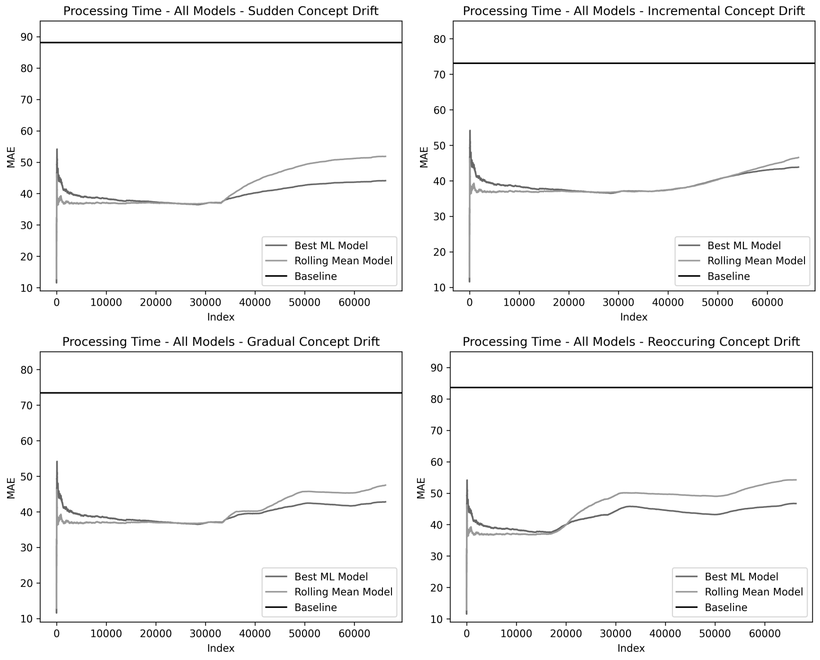

Figure 5.14: PTP performance over time of the best ML models against the RMMs and the baselines under all four concept drift sub-scenarios.

As expected, once the concept drifts emerge, the performance curves of the ML models and the RMMs start to part way. In all sub-scenarios, the RMMs are unable to adapt to the concept drift as quickly as the ML models. The largest offset between the two performance curves is observed in the sudden concept drift sub-scenario with an abrupt rise while the lowest offset is seen in the incremental concept drift sub-scenario as the change occurs slowly over time, granting both models to adapt more easily.

Both the gradual and reoccurring concept drift sub-scenarios show the performance curves bouncing up and down as the change is not occurring consistently and thus makes it more difficult for the models to adapt.

Transition Time

The results of the TTP under all four concept drift sub-scenarios are shown in Table 5.9. As stated before, there is no expert estimation baseline hence only the best ML models and RMMs are compared to one another.

Table 5.9: Best-performing model pipelines versus the RMMs and baselines for the TTP under the concept drift scenarios sudden, incremental (inc.), gradual, and reoccurring (reoc.).

(Hyper-)Parameter	**Sudden**	**Inc.**	**Gradual**	**Reoc.**
Best ML Models				
Algorithm	OXTR			
n_models	50			
max_features	25%			
max_depth	5			
min_samples_split	3			
MAE	1,148.35	1,155.62	1,113.80	1,240.27
Improvement over RMM	13.84%	02.41%	08.08%	12.88%
Rolling Mean Models (RMMs)				
MAE	1,332.78	1,184.11	1,211.72	1,423.62

The overall picture is similar to the PTP under concept drift. Improvements of the best ML models over the RMMs range from a mere 2% to over 13%. The sub-scenario where the offfset is lowest is again under incremental concept drift. Figure 5.15 supports these findings with a visualisation of the error curves over time.

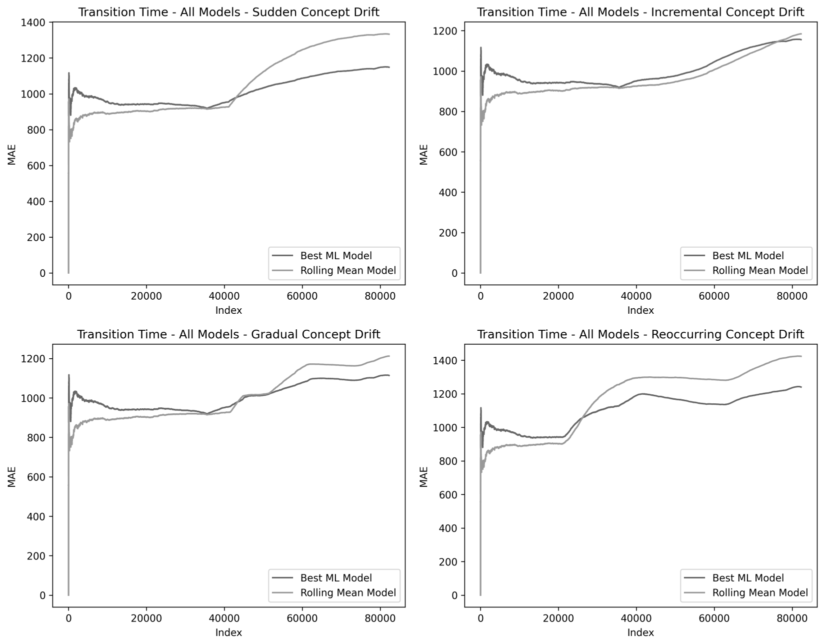

Figure 5.15: TTP performance over time of the best ML models against the RMMs and the baselines under all four concept drift sub-scenarios.

Similar to the error curves for the PTP, the MAEs spike once concept drift emerges, though the ML models adapt sooner than the RMMs. The point where the RMMs' error curves surpass the ML models' depends on the type and the magnitude of the concept drift.

5.2.4 Discussion

To summarise, case study B was designed as a close replication of case study A with one key difference: it used case study A's data and introduced four distinct types of concept drift, based on the concept drift patterns described in Section 2.2.3 and referred to as the four sub-scenarios [GŽB+14]. Then, the method blocks one and two, as well as the monitoring and maintenance steps of block three, were applied. The results were similar to the findings of case study A in that the ML models consistently outperformed expert estimation baselines as well as the RMMs. It was also found that the ML models, thanks to their adaptive nature, were able to rapidly adapt to concept drift and thus maintain performance. The speed of the adaptation was largely dictated by the concept drift pattern and magnitude.

5.3 Conclusion

In this chapter, the method laid out in Section 4.2 was applied to two case studies. Case study A used real-world data to validate the first two method blocks data preparation and model development. Case study B was built on real-world data modified by introducing artificial concept drift in order to validate the third method block model deployment and specifically its concept drift detection and adaptation mechanisms. The results were discussed individually and are now used to assess the method against the thesis' MO which Section 1.2.1 introduced as:

> MO: End-to-end methodisation of a PPR-specific LTP for SME from the MTO domain using AI.

The method covers the entire machine learning pipeline from data preparation via model development all the way to model deployment and is thus end-to-end. It enables PPR-specific LTP as the developed models are tailored toward predicting PT and TT for a given process refining a specific product on a selected resource.

AI is employed in the form of ML which is tailored toward SME from the MTO domain by utilising AutoML techniques as these are aimed at simplifying the model development and thus enable non-experts to create ML models. Following the assessment of the technical objectives in Section 4.4.1 and based on the results of the method execution within the two case studies, the MO is considered achieved to a satisfying degree.

However, there are two caveats which need to be addressed. For once, the case studies showed that, while consistent, the degree of outperformance of simpler statistical models by ML is limited. It is thus questionable whether this performance increase justifies the overhead of using the method. As analysis revealed, there are issues with the data quality in the case studies. It is thus reasonable to expect a larger degree of outperformance with an increase in data quality, rendering the cost-benefit-relation more in favour of applying the method. Thus the priority for companies seeking to apply the method should be to ensure a consistently high data quality.

The second caveat concerns the AutoML support which, while highly useful in the model development phase, overall falls behind its potential. A large portion of the effort required to apply the method falls into its first block data preparation, which is hardly supported by AutoML. Thus companies cannot rely on AutoML and domain specialists alone to apply the method but still require data science support for a successful application.

6 Summary & Outlook

What the Future Holds

This last chapter concludes this thesis. It first summarises the highlights of the previous chapters before providing an outlook into the future.

6.1 Summary

As was motivated in Chapter 1, manufacturing industries face increasingly complex customer demands in terms of product customisation and delivery times while simultaneously being met with market uncertainty and disruptions within supply networks. Especially small and medium enterprises (SME) from the make-to-order (MTO) domain seek new approaches to uphold their adherence to schedule in the light of these challenges. As such, methods for the smart scheduling of manufacturing jobs gained traction in recent years due to advances in artificial intelligence (AI), and particularly in machine learning (ML). The basis for any smart job scheduling approach however is a prediction of the lead time (LT) as this forms a crucial input variable for the job scheduling algorithm. LT is the sum of processing time (PT), i.e. the value adding time, and transition time (TT), i.e. the time spent waiting or being transported. Today, this often manual prediction lacks the necessary accuracy, especially when considering the highly customised products of the MTO domain. Factors influencing LT are manifold and can be product-, process-, or resource- (PPR-) specific. Meanwhile, the advances in ML allow for data-driven approaches to automate this prediction. On the one hand, automated ML (AutoML) techniques pose a promising research avenue as automation of complex ML engineering processes would directly empower domain specialists in the manufacturing enterprises.

On the other hand, online ML provides the ability to adapt to changes negatively impacting ML performance, such as concept drift, which are to be expected in the highly dynamic environment of a shop floor in MTO production. Therefore, this thesis set out the main objective of an end-to-end methodisation of a PPR-specific LT prediction (LTP) for SME from the MTO domain using AI in the form of AutoML as well as online ML.

Chapter 2 laid out the managerial and technical foundations for this thesis. The term production was defined and different types and organisational approaches were distinguished. Classification schemes for production types were used to highlight the MTO focus of this thesis. Five models of production organisation were discussed an the job shop production model was highlighted as most relevant for this thesis. Concluding the production section, the concept, goals, and system landscape of PPC were introduced. Furthermore, the foundations of AI as relevant to this thesis were discussed. Several definitions of AI were provided before settling for ML as the main AI-application domain. This was briefly defined and further systemised by highlighting both the types of learning, with supervised learning identified as the most relevant in the context of this work, as well as introducing applicable learning techniques, such as online learning. Considerations were given on different feature engineering techniques for both offline and online learning. Then the problem of concept drift in ML was introduced, the four archetypical concept drift patterns were discussed, and various applicable detection and adaptation methods were presented. With the ML basics covered, AutoML and its methods as well as concrete solutions were introduced as means to automate steps of the laborious ML engineering process. Concluding the chapter, a brief overview of process models for ML engineering was given which serve as a framework for the method developed in this thesis.

Chapter 3 provided an in-depth dissemination of the state of the art in ML-supported LTP. After further systematising the problem of LTP into regression and classification modelling approaches, the scope and structure of the successive literature review was defined. It was found that the existing body of work could be distinguished into papers on combined LTP and PT prediction as opposed to works focusing on TT prediction. As a trend, early works experimented with various ML algorithms and established the general feasibility of applying an ML approach to LTP.

Later works however revolved more around ML engineering approaches and system architectures as well as integration into PPC. The studies were summarised and compared based on their content along the categories production domains (mostly Batch / MTO / SS), data sources (real-world / simulation), problem types (classification / regression), algorithms, and key results. Concluding the chapter, several research gaps were derived to guide the work presented in this thesis. These highlight the need for a holistic approach to LTP while addressing generally poor model generalisation and adaptation as well as simplifying the model development process, rendering it more accessible for SMEs.

Chapter 4 described the main contribution of this thesis: the method for AutoML-supported LTP. It began by laying out the framework in which the method is meant to be operated. As such, the main objective was further systemised into technical objectives. The data streaming scenario was outlined and preconditions for the method application were provided. This was followed by an in-depth description of the method which is organised in two stages, protoyping and production, dividing into the three blocks data preparation, model development, and model deployment. These three blocks are subdivided into a total of twelve steps of which each provides concrete guidelines and makes suggestions on suitable methods for implementation. Subsequently, system architectures for inference and training pipelines based on the streaming scenario and the method steps were illustrated. Concluding the chapter, the designed method was successfully assessed against the technical objectives. Furthermore, the AutoML support as provided by the state of the art was assessed based on its ability to cover each method step. This yielded mixed results, thus highlighting potential research gaps. Finally, the method presented in this thesis was sufficiently distinguished from the existing methods for LTP.

Chapter 5 validated the method in two case studies. Case study A was based on a real-world dataset from an SME operating in an MTO fashion. Case study B altered the real-world dataset by introducing artificial concept drift in order to assess the method's capabilities to prevent performance degradation under such circumstances. The resulting LTP models were benchmarked against baseline expert estimations, available for PT only, as well as simple statistical models. For case study A, it was found that the LTP models' errors were reduced by 35% over the expert baseline and 3% to 6% over the statistical models.

In case study B, depending on the type of concept drift, improvements ranged from 38% to 50% over the baseline and 2% to 15% over the statistical models respectively. Further findings included a large variety in error magnitudes depending on the underlying production processes as for some LTs are more difficult to predict than for others. While the LTP models generated by the method presented in this thesis already consistently outperform both the expert baseline as well as the statistical models, and are capable of addressing concept drift, it is expected that improvements in data quality will render these models more effective in order to leverage the full potential.

6.2 Outlook

With an ongoing digitisation in SMEs in general, and on the shop floor in particular, higher volume and variety of structured data are expected to be available in the future. With the shift away from paper-based job scheduling and since manual data inputs have been identified as a key source of data errors in the case studies, further automation of the job scheduling and execution monitoring will lead to an increase in overall data quality. In turn, this will render the application of the method more attractive as prediction errors decline.

Advancements toward improving and broadening of AutoML techniques form another avenue of improvement. Namely refining interpretability and robustness could have a positive effect on the model performance. This can be expanded upon by supporting other tasks in the ML pipeline outside of the model development, such as data preparation and model maintenance, in order to achieve true end-to-end AutoML support. These improvements would empower especially SME to apply more data-driven analytics and optimisations, such as the method presented in this thesis, to their operations.

Lastly, the advent of generative AI in the form of large language models (LLM) opens an interesting direction for future research. These could help bridge the gap between domain specialists' needs and AutoML solutions' functionalities by actively supporting labour-intense or difficult method steps. For example, users could request analysis or transformations from LLMs in their own domain language. This would provide another cornerstone in the empowerment of SME to develop their own customised AI solutions.

A Appendix

A.1 Supplements for Case Study A

A.1.1 Analysis

Table A.1: Column descriptions of the raw data.

Column	**Data Type**	**Remark**
JOB_ID	Numeric	Distinctly identifies a job.
JOB_DESCRIPTION	String	Descriptive characterisation for each job.
PART_DESCRIPTION	String	Additional information on the product and processing step. Six different values.
CUSTOMER_ID	String	Distinctly identifies a customer.
PRIO_CLASS_1-4	Boolean	Four company-specific prioritisation flags.
MATERIAL_DESCRIPTION	String	European standard (EN) material number.
MATERIAL_FORMING_TEMP	String	Max-Min forming temperatures represented as a combined string.
MATERIAL_SPECIFIC_WEIGHT	Numeric	Specific weight of the material.
DATE_REGISTERED	Date	Date of job registration.
DATE_APPROVED	Date	Date of approval (release) for production.
DATE_PLANNED_DELIVERY_INTERNAL	Date	Internal due date of the job.
DATE_PLANNED_DELIVERY_EXTERNAL	Date	Customer delivery date of the job.
MACHINE_GROUP_ID	Numeric	Distinctly identifies a machine group.
MACHINE_GROUP_LOAD	Numeric	Load factor of the machine group at the time of reporting. At times > 1.
MACHINE_GROUP_ATTENDANCE_RATE	Numeric	Staff availability [0, 1] of the machine group at the time of reporting.
JOB_ESTIMATED_LEAD_TIME	Numeric	Estimated lead time of the job in days.
PROCESS_ID	Numeric	Distinctly identifies a type of process.
PROCESS_DESCRIPTION	String	Describes a type of process.
PROCESS_ESTIMATED_PROCESSING_ TIME_MANUAL	**Numeric**	**Estimated processing time of this operation. Baseline variable.**
WORK_SEQUENCE_ID	Numeric	Planned position in the work sequence.
RUN_ID	Numeric	Reported position in the work sequence.
DATE_FROM	Datetime	Reported start datetimestamp.
DATE_TO	Datetime	Reported end datetimestamp.
PROCESS_ACTUAL_PROCESSING_TIME	**Numeric**	**Reported processing time of this operation. Target variable.**
PROCESS_ACTUAL_TRANSITION_TIME	**Numeric**	**Reported transition time between this and the previous operation. Target variable.**
ACTUAL_TO_ESTIMATED_PROCESSING_TIME	Numeric	A posteriori [0, 1] relationship between the estimated and actual processing times.
ADHERENCE_TO_SCHEDULE	Numeric	A posteriori [0, 1] measuring if the operation was executed on time.

Table A.2: Amount of NaN-values in the raw data. Columns not listed showed no NaN-values.

Column	NaN-Values
PART_DESCRIPTION	2
MATERIAL_DESCRIPTION	7
MATERIAL_FORMING_TEMP	4,757
DATE_PLANNED_DELIVERY_INTERNAL	38,158

Table A.3: Categorical columns in the raw data. Columns not listed were not deemed categorical.

Column	Categories
PART_DESCRIPTION	6
CUSTOMER_ID	641
MATERIAL_DESCRIPTION	601
MACHINE_GROUP_ID	26
PROCESS_ID	22
PROCESS_DESCRIPTION	22

Table A.4: Frequency-based percentage shares of all processes executed in the raw data.

PROCESS_DESCRIPTION	Share (%)
Sawing	17.8255
Forging	13.2578
Final Control & Packaging	12.4359
Measurement Control Marking	11.6643
Hardness Testing	9.8175
Rolling	7.9668
Preturning	7.8050
Setting Up	4.7204
Testing for Mixed-Up Components	3.7458
Finish Turning	2.5702
Milling	2.4466
Reworking	1.8930
Sampling	1.8478
Turning	1.3795
Waiting for Material	0.2492
Waiting for QA	0.1999
Drilling	0.1135
Maintaining	0.0271
Centering	0.0161
Downtime / Repairing	0.0141
Machining	0.0030
Sawing Endings	0.0010

Table A.5: PT-based percentage shares of all processes executed in the raw data.

PROCESS_DESCRIPTION	**Share (%)**
Preturning	27.847299
Sawing	23.720506
Milling	15.955609
Finish Turning	9.925687
Forging	4.138404
Measurement Control Marking	2.939754
Setting Up	2.843955
Turning	2.554872
Sampling	2.202137
Hardness Testing	2.121552
Final Control & Packaging	1.910601
Reworking	1.635673
Rolling	0.997453
Testing for Mixed-Up Components	0.767323
Drilling	0.155988
Waiting for QA	0.148877
Waiting for Material	0.069273
Machining	0.033440
Maintaining	0.015723
Downtime / Repairing	0.008433
Centering	0.007440
Sawing Endings	0.000000

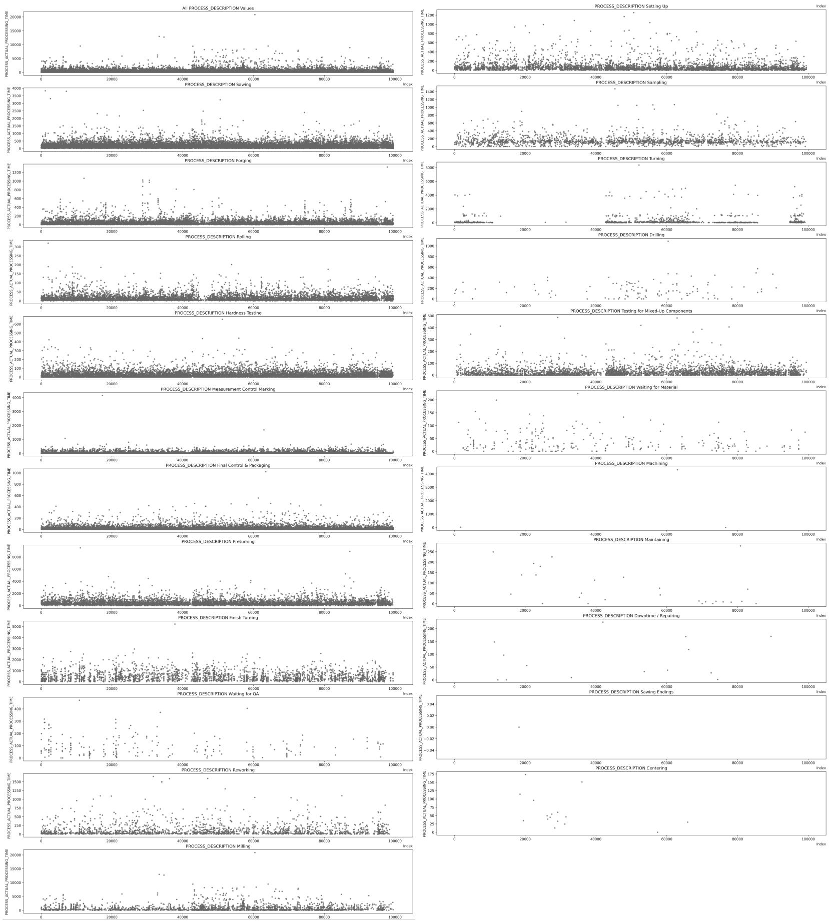

Figure A.1: PTs per process over time in the raw data.

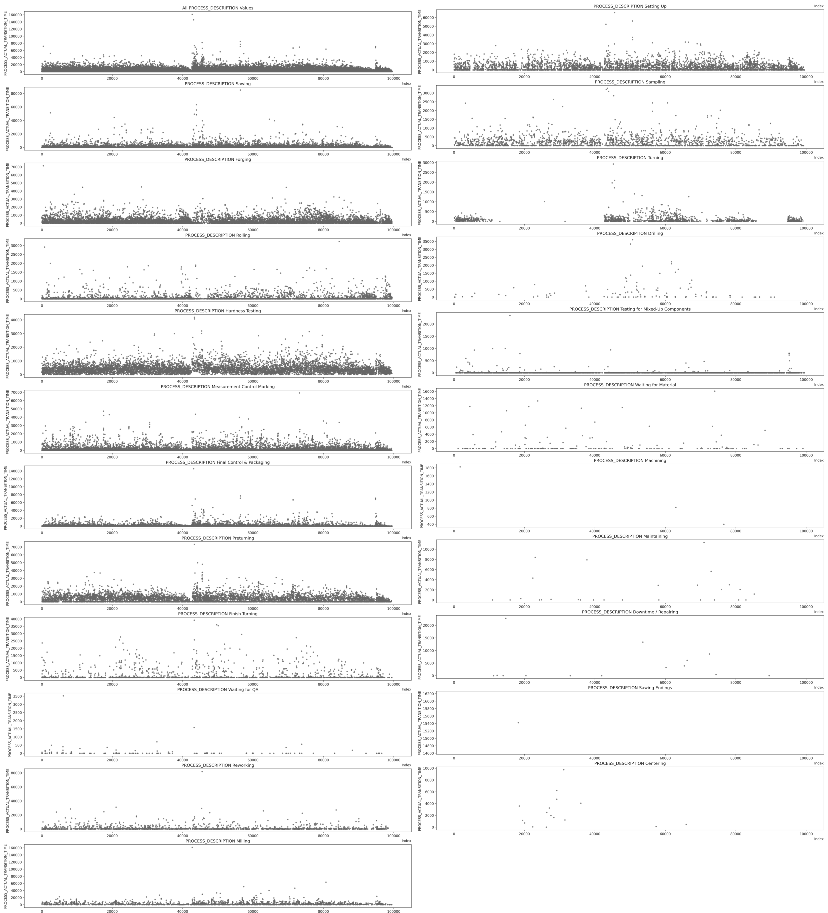

Figure A.2: TTs per process over time in the raw data.

A.1.2 Transformation

Table A.6: Column descriptions of the transformed data. Xs highlight which columns have been additionally joined from previous operations of the respective job and machine group.

Column	**Prev. Op. Job**	**Prev. Op. M-Grp.**	**Remark**
JOB_ID		X	Distinctly identifies a job.
JOB_DESCRIPTION		X	Descriptive characterisation for each job.
PART_DESCRIPTION		X	Additional information on the product and processing step. Six different values.
CUSTOMER_ID		X	Distinctly identifies a customer.
CUSTOMER_CLASS		X	Class derived from the customer id pattern.
PRIO_CLASS_1-4		X	Four company-specific prioritisation flags.
MATERIAL_DESCRIPTION		X	European standard (EN) material number.
MATERIAL_TYPE		X	Main material group.
MATERIAL_SUB_TYPE		X	Sub material group.
MATERIAL_SUB_TYPE_INDEX		X	Sub material group counter.
MATERIAL_FORMING_TEMP_MIN		X	Min forming temperature.
MATERIAL_FORMING_TEMP_MAX		X	Max forming temperature.
MATERIAL_FORMING_TEMP_MIN_MAX_DELTA		X	Distance between min and max forming temperature.
MATERIAL_SPECIFIC_WEIGHT		X	Specific weight of the material.
DATE_REGISTERED		X	Date of job registration.
DATE_APPROVED		X	Date of approval (release) for production.
DATE_PLANNED_DELIVERY_INTERNAL		X	Internal due date of the job.
DATE_PLANNED_DELIVERY_EXTERNAL		X	Customer delivery date of the job.
DATE_REGISTERED_APPROVED_DELTA		X	Distance between job registration and release.
DATE_REGISTERED_PLANNED_INTERNAL_DELTA		X	Distance between job registration and internal due date.
DATE_APPROVED_PLANNED_INTERNAL_DELTA		X	Distance between job release and internal due date.
DATE_REGISTERED_PLANNED_EXTERNAL_DELTA		X	Distance between job registration and customer delivery date.
DATE_APPROVED_PLANNED_EXTERNAL_DELTA		X	Distance between job release and customer delivery date.
DATE_PLANNED_DELIVERY_INTERNAL_EXTERNAL_DELTA		X	Distance between internal due date and customer delivery date.
MACHINE_GROUP_ID	X		Distinctly identifies a machine group.
MACHINE_GROUP_LOAD	X		Load factor of the machine group at the time of reporting. At times > 1.
MACHINE_GROUP_ATTENDANCE_RATE	X		Staff availability [0, 1] of the machine group at the time of reporting.
JOB_ESTIMATED_LEAD_TIME		X	Estimated lead time of the job.
PROCESS_ID		X	Distinctly identifies a type of process.
PROCESS_DESCRIPTION		X	Describes a type of process.
PROCESS_ESTIMATED_PROCESSING_TIME_MANUAL	X	X	**Estimated processing time of this operation. Baseline variable.**
WORK_SEQUENCE_ID	X	X	Planned position in the work sequence.
RUN_ID	X	X	Reported position in the work sequence.
DATE_FROM	X	X	Reported start datetimestamp.
DATE_TO	X	X	Reported end datetimestamp.
PROCESS_ACTUAL_PROCESSING_TIME	X	X	**Reported processing time of this operation. Target variable.**
PROCESS_ACTUAL_TRANSITION_TIME	X	X	**Reported transition time between this and the previous operation. Target variable.**
ACTUAL_TO_ESTIMATED_PROCESSING_TIME	X	X	A posteriori [0, 1] relationship between the estimated and actual processing times.
ADHERENCE_TO_SCHEDULE	X	X	A posteriori [0, 1] measuring if the operation was executed on time.

A.1.3 Filtering

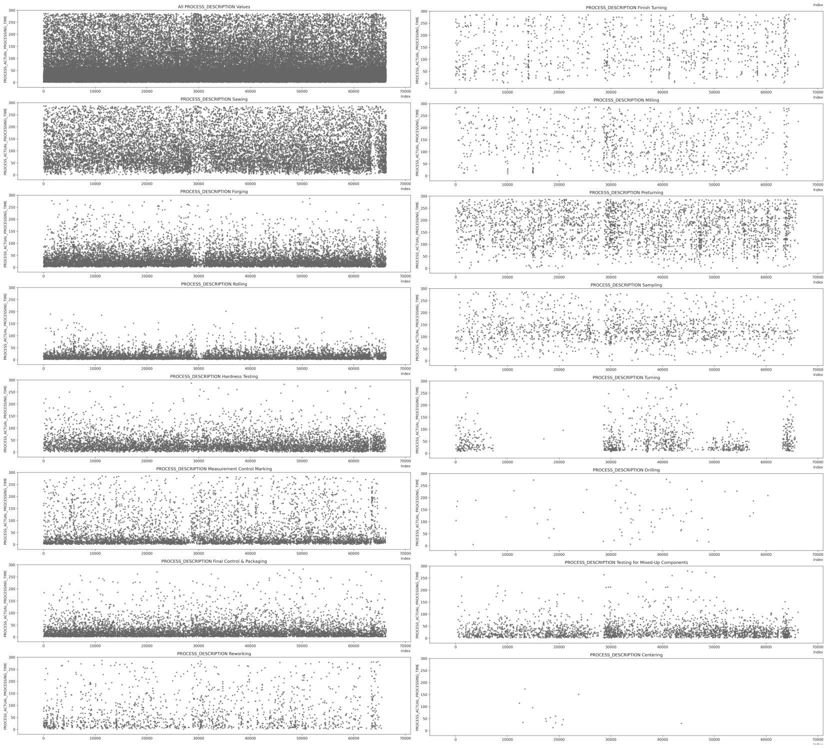

Figure A.3: PTs per process over time in the filtered data.

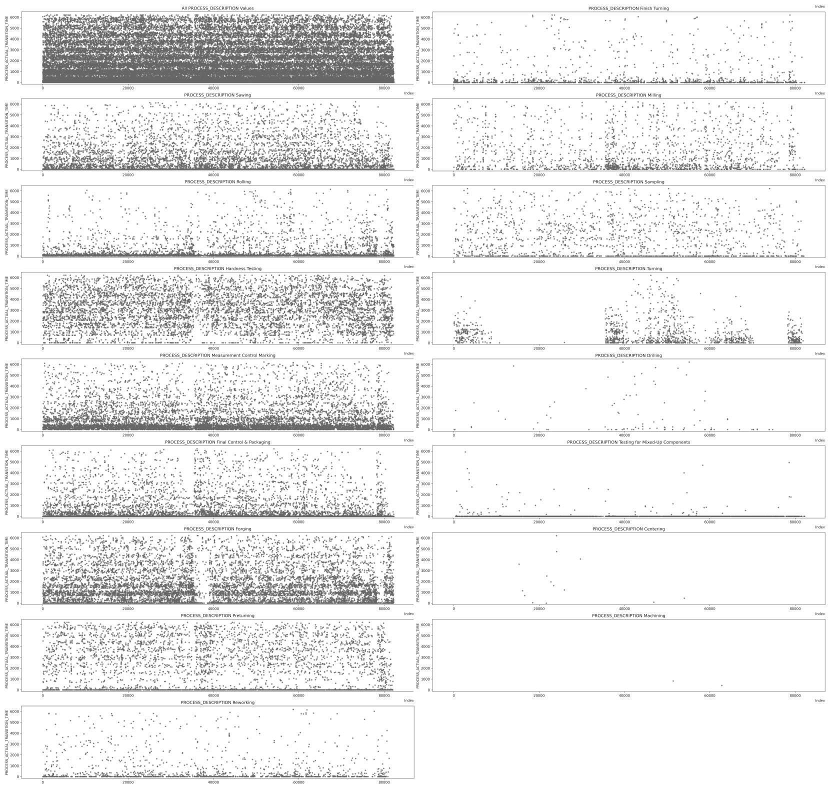

Figure A.4: TTs per process over time in the filtered data.

A.1.4 Selection

Table A.7: Column descriptions of the selected data. Xs highlight which columns have been additionally joined from previous operations of the respective job and machine group.

Column	Prev. Op. Job	Prev. Op. M-Grp.	Remark
PART_DESCRIPTION		X	Additional information on the product and processing step. Six different values.
CUSTOMER_ID		X	Distinctly identifies a customer.
CUSTOMER_CLASS		X	Class derived from the customer id pattern.
PRIO_CLASS_1-4		X	Four company-specific prioritisation flags.
MATERIAL_DESCRIPTION		X	European standard (EN) material number.
MATERIAL_TYPE		X	Main material group.
MATERIAL_SUB_TYPE		X	Sub material group.
MATERIAL_SUB_TYPE_INDEX		X	Sub material group counter.
MATERIAL_FORMING_TEMP_MIN		X	Min forming temperature.
MATERIAL_FORMING_TEMP_MAX		X	Max forming temperature.
MATERIAL_FORMING_TEMP_MIN_MAX_DELTA		X	Distance between min and max forming temperature.
MATERIAL_SPECIFIC_WEIGHT		X	Specific weight of the material.
DATE_REGISTERED_APPROVED_DELTA		X	Distance between job registration and release.
DATE_REGISTERED_PLANNED_INTERNAL_DELTA		X	Distance between job registration and internal due date.
DATE_APPROVED_PLANNED_INTERNAL_DELTA		X	Distance between job release and internal due date.
DATE_REGISTERED_PLANNED_EXTERNAL_DELTA		X	Distance between job registration and customer delivery date.
DATE_APPROVED_PLANNED_EXTERNAL_DELTA		X	Distance between job release and customer delivery date.
DATE_PLANNED_DELIVERY_INTERNAL_EXTERNAL_DELTA		X	Distance between internal due date and customer delivery date.
MACHINE_GROUP_ID	X		Distinctly identifies a machine group.
MACHINE_GROUP_LOAD	X		Load factor of the machine group at the time of reporting. At times > 1.
MACHINE_GROUP_ATTENDANCE_RATE	X		Staff availability [0, 1] of the machine group at the time of reporting.
PROCESS_ID		X	Distinctly identifies a type of process.
WORK_SEQUENCE_ID	X	X	Planned position in the work sequence.
RUN_ID	X	X	Reported position in the work sequence.

A.1.5 Configuration

Table A.8: Pool of important hyperparameters for the configuration step for the ARFR [GBR+17] and OXTR [MNVdC22] algorithms.

Parameter	Value Pool
preprocessing	Feature Scaling
n_models	10, 25, 50, 75, 100
max_features	25%, 50%, 75%, 100%, sqrt, log2
metric	MAE
max_depth	5, 10, Unlimited
min_samples_split	3, 5, 10
drift_detector	ADWIN

A.2 Supplement for Case Study B

A.2.1 Scenario

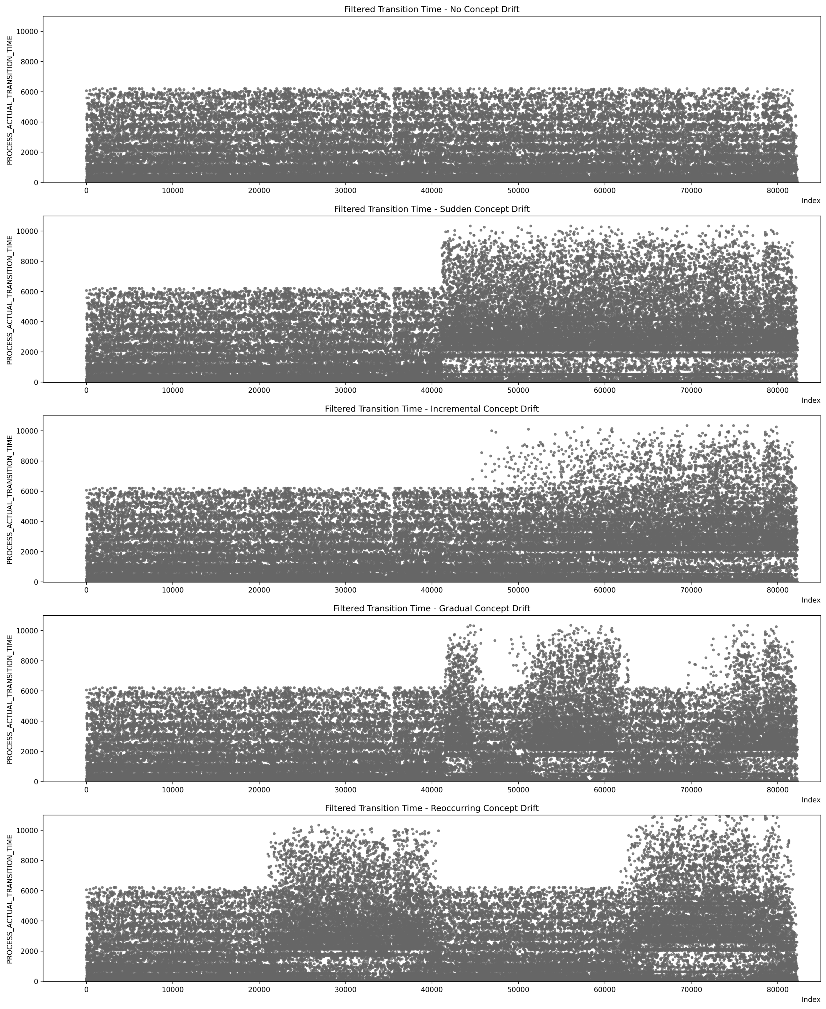

Figure A.5: The original filtered TTs in comparison to the altered datasets of all four concept drift patterns.

Publications

Table A.9: List of publications.

Year	Title	Authors	Reference
2015	A Comparison of Agent-Based Coordination Architecture Variants for Automotive Product Change Management	Janek Bender, Stefan Kehl, Jörg P. Müller	[BKM15]
2019	Closed-Loop-Engineering – Enabler for Swift Reconfiguration in Plant Engineering	Janek Bender, Jana Deckers, Simon Fritz, Jivka Ovtcharova	[BDFO19]
2019	VR-gestütztes Closed-Loop-Engineering für schnelle Rekonfigurationsprozesse im Anlagenbau	Janek Bender, Jana Deckers, Lucas Kirsch, Jivka Ovtcharova	[BDKO19]
2021	Prototyping Machine-Learning-Supported Lead Time Prediction Using AutoML	Janek Bender, Jivka Ovtcharova	[BO21]
2022	Benchmarking AutoML-Supported Lead Time Prediction	Janek Bender, Martin Trat, Jivka Ovtcharova	[BTO22]
2022	Unsupervised Anomaly Detection and Root Cause Analysis for an Industrial Press Machine based on Skip-Connected Autoencoder	Chenwei Sun, Martin Trat, Janek Bender, Jivka Ovtcharova, George Jeppesen, Jan Bär	[STB+22]
2023	Towards a B2B integration framework for smart services in Industry 4.0	Viktor Schubert, Steffen Kuehner, Tobias Krauss, Martin Trat, Janek Bender	[SKK+23]

2023	Sensitivity-Based Optimization of Unsupervised Drift Detection for Categorical Data Streams	Martin Trat, Janek Bender, Jivka Ovtcharova	[TBO23]
2023	Energy-Flexible Job-Shop Scheduling Using Deep Reinforcement Learning	Mine Felder, Daniel Steiner, Paul Busch, Martin Trat, Chenwei Sun, Janek Bender, Jivka Ovtcharova	[FSB$^+$23]
2024	Artificial-intelligence-enabled dynamic demand response system for maximizing the use of renewable electricity in production processes	Hendro Wicaksono, Martin Trat, Bashyal Atit, Tina Boroukhian, Mine Felder, Mischa Ahrens, Janek Bender, Sebastian Groß, Daniel Steiner, Christoph July, Christoph Dorus, Thorsten Zoerner	[WTB$^+$24]

References

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2015. Available at: http://download.tensorflow.org/paper/whitepaper2015.pdf. Last visited on 14.12.2023.

[AAH08] Mouhib Alnoukari, Zaidoun Alzoabi, and Saiid Hanna. Applying adaptive software development (ASD) agile modeling on predictive data mining applications: ASD-DM methodology. In *2008 International Symposium on Information Technology*, pages 1–6. IEEE, 2008.

[AG19] Plamen P. Angelov and Xiaowei Gu. *Empirical Approach to Machine Learning*, volume 800. Springer International Publishing, Cham, 2019.

[ALAMM+18] Santiago Angée, Silvia I. Lozano-Argel, Edwin N. Montoya-Munera, Juan-David Ospina-Arango, and Marta S. Tabares-Betancur. Towards an Improved ASUM-DM Process Methodology for Cross-Disciplinary Multi-organization Big Data & Analytics Projects. In Lorna Uden, Branislav Hadzima, and I-Hsien Ting, editors, *Knowledge Management in Organizations*, volume 877 of *Communications in Computer and Infor-*

mation Science, pages 613–624. Springer International Publishing, Cham, 2018.

[All01] Paul D. Allison. *Missing Data*. Quantitative Applications in the Social Sciences Ser. SAGE Publications, Incorporated, Thousand Oaks, 2001.

[Ama22] Amazon.com, Inc. Amazon SageMaker Autopilot, 2022. Available at: https://aws.amazon.com/sagemaker/autopilot/. Last visited on 19.08.2022.

[AMT08] Abdulrahman Alenezi, Scott A. Moses, and Theodore B. Trafalis. Real-time prediction of order flowtimes using support vector regression. *Computers & Operations Research*, 35(11):3489–3503, 2008.

[BB12] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

[BBO18] BBOB. Black-box Optimization Benchmarking (BBOB) workshop series, 2018. Available at: http://numbbo.github.io/workshops/index.html. Last visited on 13.05.2022.

[BCdF10] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning, 2010. Available at: http://arxiv.org/pdf/1012.2599v1.

[BDFO19] Janek Bender, Jana Deckers, Simon Fritz, and Jivka Ovtcharova. Closed-Loop-Engineering – Enabler for Swift Reconfiguration in Plant Engineering. In *Proceedings of the European Modeling and Simulation Symposium, 2019*, pages 138–144. 2019.

[BDKO19] Janek Bender, Jana Deckers, Lucas Kirsch, and Jivka Ovtcharova. VR-gestütztes Closed-Loop-Engineering für schnelle Rekonfigurationsprozesse im Anlagenbau. In *Tagungsband VAR^2 2019 – Realität erweitern*, pages 273–282. 2019.

[Bel03] Richard Ernest Bellman. *Dynamic programming*. Dover Publications, Mineola, N.Y, dover ed. edition, 2003.

[Ben09] Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

[BG07] Albert Bifet and Ricard Gavaldà. Learning from Time-Changing Data with Adaptive Windowing. In Chid Apte, David Skillicorn, Bing Liu, and Srinivasan Parthasarathy, editors, *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448, Philadelphia, PA, 04262007. Society for Industrial and Applied Mathematics.

[BGDF+06] Manuel Baena-Garcıa, José Del Campo-Ávila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86, 2006.

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. Computer science. Springer, New York, NY, 2006.

[BKM15] Janek Bender, Stefan Kehl, and Jörg P. Müller. A Comparison of Agent-Based Coordination Architecture Variants for Automotive Product Change Management. In Jörg P. Müller, Wolf Ketter, Gal Kaminka, Gerd Wagner, and Nils Bulling, editors, *Multiagent System Technologies*, volume 9433 of *Lecture Notes in Computer Science*, pages 249–267. Springer International Publishing, Cham, 2015.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[BO21] Janek Bender and Jivka Ovtcharova. Prototyping Machine-Learning-Supported Lead Time Prediction Using AutoML. *Procedia Computer Science*, 180:649–655, 2021.

[BPZL12] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *The Journal of Machine Learning Research*, 13:27–66, 2012.

[Bre01] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[BTO22] Janek Bender, Martin Trat, and Jivka Ovtcharova. Benchmarking AutoML-Supported Lead Time Prediction. *Procedia Computer Science*, 200:482–494, 2022.

[Cad17] Field Cady. *The Data Science Handbook*. John Wiley & Sons, Inc, Hoboken, New Jersey, 2017.

[CCK+00] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. CRISP-DM 1.0: Step-by-step data mining guide. 2000.

[CHH02] Murray Campbell, A. Joseph Hoane, and Feng-hsiung Hsu. Deep Blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

[Cho15] François Chollet. Keras, 2015. Available at: https://github.com/fchollet/keras. Last visited on 16.08.2022.

[Cle11] Thomas Cleff. *Deskriptive Statistik und moderne Datenanalyse: Eine computergestützte Einführung mit Excel, PASW (SPSS) und STATA*. Lehrbuch. Gabler Verl. Springer Fachmedien, Wiesbaden, 2., überarb. und erw. aufl. edition, 2011.

[Cox72] David R. Cox. Regression Models and Life-Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972.

[CZ13] Edwin K. P. Chong and Stanislaw H. Zak. *An Introduction to Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2013.

[dCGMT10] Francisco Javier de Cos Juez, Paulino Jose García Nieto, Javier Martínez Torres, and Javier Taboada Castro. Analysis of lead times of metallic components in the aerospace industry through a supported vector machine model. *Mathematical and Computer Modelling*, 52(7-8):1177–1184, 2010.

[DH00] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD interna-*

tional conference on Knowledge discovery and data mining, pages 71–80, 2000.

[DHM+20] Atsushi Deguchi, Chiaki Hirai, Hideyuki Matsuoka, Taku Nakano, Kohei Oshima, Mitsuharu Tai, and Shigeyuki Tani. What Is Society 5.0? In *Society 5.0*, pages 1–23. Springer Singapore, Singapore, 2020.

[DIN15] DIN EN 10027-2:2015-07, Bezeichnungssysteme für Stähle_-Teil_2: Nummernsystem; Deutsche Fassung EN_10027-2:2015, 2015.

[DSH15] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. In *IJCAI*, 2015.

[EFH+20] Marlene Eisenträger, Christian Frey, Andreas Herzog, Ali Moghiseh, Lukas Morand, Julius Pfrommer, Henrike Stephani, Anke Stoll, and Lars Wessels. ML4P: Vorgehensmodell Machine Learning for Production, 2020.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[EMH19] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *J. Mach. Learn. Res.*, 20(1):1997–2017, 2019.

[EMS09] Hugo Jair Escalante, Manuel Montes, and Luis Sucar. Particle Swarm Model Selection. *Journal of Machine Learning Research*, 10:405–440, 2009.

[Eve02] Walter Eversheim. *Organisation in der Produktionstechnik 3*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[FHW16] Eibe Frank, Mark A. Hall, and Ian H. Witten. The WEKA Workbench. In *Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. 2016.

[Fie00] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[FKE+15] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and Robust Automated Machine Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2755–2763, Cambridge, MA, USA, 2015. MIT Press.

[FKH18] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446. PMLR, 2018.

[Fra15] Stuart Frankel. Data Scientists Don't Scale, 2015. Available at: https://hbr.org/2015/05/data-scientists-dont-scale. Last visited on 30.12.2021.

[FSB+23] Mine Felder, Daniel Steiner, Paul Busch, Martin Trat, Chenwei Sun, Janek Bender, and Jivka Ovtcharova. *Energy-Flexible Job-Shop Scheduling Using Deep Reinforcement Learning*. Hannover : publish-Ing, 2023.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts and London, England, 2016.

[GBE+22] Josef Gramespacher, Wener Breisacher, Alexander Essig, Janek Bender, and Matthias Gloß. Alto - Algorithmengestützte Optimierung der termingerechten Auftragssteuerung für die Organisationsabläufe der Einzelfertigung. *TIB - Technische Informationsbibliothek Universitätsbibliothek Hannover*, 2022.

[GBR+17] Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.

[GE06] Isabelle Guyon and André Elisseeff. An Introduction to Feature Extraction. In Isabelle Guyon, Masoud Nikravesh, Steve Gunn, and Lotfi A. Zadeh, editors, *Feature Extraction*, volume

207 of *Studies in Fuzziness and Soft Computing*, pages 1–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[GLH11] Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized Fisher Score for Feature Selection. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, pages 266–273, Arlington, Virginia, USA, 2011. AUAI Press.

[GMCR04] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with Drift Detection. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Ana L. C. Bazzan, and Sofiane Labidi, editors, *Advances in Artificial Intelligence – SBIA 2004*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Goo22] Google LLC. Google Cloud AutoML, 2022. Available at: https://cloud.google.com/automl. Last visited on 19.08.2022.

[GPBG18] Dávid Gyulai, András Pfeiffer, Júlia Bergmann, and Viola Gallina. Online lead time prediction supporting situation-aware production control. *Procedia CIRP*, 78:190–195, 2018.

[GPN$^+$18] Dávid Gyulai, András Pfeiffer, Gábor Nick, Viola Gallina, Wilfried Sihn, and László Monostori. Lead time prediction in a flow-shop environment with analytical and machine learning approaches. *IFAC-PapersOnLine*, 51(11):1029–1034, 2018.

[GRB$^+$19] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 21(2):6–22, 2019.

[GSTH14] Peter Greschke, Malte Schönemann, Sebastian Thiede, and Christoph Herrmann. Matrix Structures for High Volumes and Flexibility in Production Systems. *Procedia CIRP*, 17:160–165, 2014.

[GŽB$^+$14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.

[HB16] Daniel Horn and Bernd Bischl. Multi-objective parameter configuration of machine learning algorithms using model-based optimization. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

[HKV19] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning*. Springer International Publishing, Cham, 2019.

[HS00] Wallace J. Hopp and Melanie L. Roof Sturgis. Quoting manufacturing due dates subject to a service level constraint. *IIE Transactions*, 32(9):771–784, 2000.

[HSP$^+$21] Constanze Hasterok, Janina Stompe, Julius Pfrommer, Thomas Usländer, Jens Ziehn, Sebastian Reiter, Michael Weber, and Till Riedel. PAISE®: Das Vorgehensmodell für KI-Engineering, 2021.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer series in statistics. Springer, New York, second edition edition, 2009.

[Hun07] John D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[HZC21] Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[Ige05] Christian Igel. Multi-objective Model Selection for Support Vector Machines. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 534–546. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[Imb20] Alexandru-Ionut Imbrea. An empirical comparison of automated machine learning techniques for data streams, 2020. Available at: http://essay.utwente.nl/80548/.

[Joy21] James Joyce. Bayes' Theorem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, 2021.

[JSH19] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-Keras: An Efficient Neural Architecture Search System. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956, New York, NY, USA, 2019. ACM.

[KKS13] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost Optimal Exploration in Multi-Armed Bandits. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1238–1246, Atlanta, Georgia, USA, 2013. PMLR.

[KLL22] Florian Kellner, Bernhard Lienland, and Maximilian Lukesch. *Produktionswirtschaft*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2022.

[KMB$^+$22] Cedric Kulbach, Jacob Montiel, Maroua Bahri, Marco Heyden, and Albert Bifet. Evolution-Based Online Automated Machine Learning. In João Gama, Tianrui Li, Yang Yu, Enhong Chen, Yu Zheng, and Fei Teng, editors, *Advances in Knowledge Discovery and Data Mining*, volume 13280 of *Lecture Notes in Computer Science*, pages 472–484. Springer International Publishing, Cham, 2022.

[KPB15] Tammo Krueger, Danny Panknin, and Mikio Braun. Fast Cross-Validation via Sequential Testing. *J. Mach. Learn. Res.*, 16(1):1103–1155, 2015.

[KSS16] Gilad Katz, Eui Chul Richard Shin, and Dawn Song. ExploreKit: Automatic Feature Generation and Selection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 979–984. IEEE, 2016.

[Kuh20] Max Kuhn. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Chapman and Hall/CRC Data Science Ser. CRC Press LLC, Milton, 2020.

[KV15] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.

[Lan01] Doug Laney. 3D Data Management: Controlling Data Volume, Velocity, and Variety: Techreport, 2001.

[LCW+18] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature Selection: A Data Perspective. *ACM Computing Surveys*, 50(6):1–45, 2018.

[LGA+18] Lukas Lingitz, Viola Gallina, Fazel Ansari, Dávid Gyulai, András Pfeiffer, Wilfried Sihn, and László Monostori. Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer. *Procedia CIRP*, 72:1051–1056, 2018.

[LH07] Shane Legg and Marcus Hutter. A Collection of Definitions of Intelligence. *Frontiers in Artificial Intelligence and Applications*, 2007.

[LH16] Ilya Loshchilov and Frank Hutter. CMA-ES for Hyperparameter Optimization of Deep Neural Networks, 2016. Available at: http://arxiv.org/pdf/1604.07269v1.

[Lit61] John D. C. Little. A Proof for the Queuing Formula. *Operations Research*, 9(3):383–387, 1961.

[Lit12] Roderick J. A. Little. *Statistical analysis with missing data*. Wiley Blackwell, Chichester, West Sussex, 3. rev. ed. edition, 2012.

[LJD+17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, 2017.

[LP20] Erin LeDell and Sebastien Poirier. H2O AutoML: Scalable Automatic Machine Learning. *7th ICML Workshop on Automated Machine Learning (AutoML)*, 2020.

[LS95] Huan Liu and Rudy Setiono. Chi2: feature selection and discretization of numeric attributes. In *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, pages 388–391. IEEE Comput. Soc. Press, 1995.

[Lug09] George F. Luger. *Artificial intelligence: Structures and strategies for complex problem solving*. Pearson/Addison-Wesley, Boston, Mass., 6. ed., pearson international ed. edition, 2009.

[LUTG17] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *The Behavioral and brain sciences*, 40:e253, 2017.

[LYS19] Zhong Heng Lim, Umi Kalsom Yusof, and Haziqah Shamsudin. Manufacturing Lead Time Classification Using Support Vector Machine. In Halimah Badioze Zaman, Alan F. Smeaton, Timothy K. Shih, Sergio Velastin, Tada Terutoshi, Nazlena Mohamad Ali, and Mohammad Nazir Ahmad, editors, *Advances in Visual Informatics*, volume 11870 of *Lecture Notes in Computer Science*, pages 268–278. Springer International Publishing, Cham, 2019.

[LYWF15] Minqi Li, Feng Yang, Hong Wan, and John W. Fowler. Simulation-based experimental design and statistical modeling for lead time quotation. *Journal of Manufacturing Systems*, 37:362–374, 2015.

[MDF$^+$14] Dimitris A. Mourtzis, Michael Doukas, Katerina Fragou, Konstantinos Efthymiou, and V. Matzorou. Knowledge-based Estimation of Manufacturing Lead Time for Complex Engineered-to-order Products. *Procedia CIRP*, 17:499–504, 2014.

[Mel66] P. Mellor. A Review of Job Shop Scheduling. *OR*, 17(2):161, 1966.

[MHM$^+$21] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in Python. *Journal of Machine Learning Research*, 22(110):1–8, 2021.

[Mic22] Microsoft Corporation. Microsoft Azure Automated Machine Learning, 2022. Available at: https://azure.microsoft.com/services/machine-learning/automatedml/. Last visited on 19.08.2022.

[Mit97] Tom M. Mitchell. *Machine learning*. McGraw-Hill series in computer science. WCB/McGraw-Hill, Boston, Mass., 1997.

[ML93] Bart L. MacCarthy and Jiyin Liu. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International journal of production research*, 31(1):59–79, 1993.

[MLRH11] Yair Meidan, Boaz Lerner, Gad Rabinowitz, and Michael Hassoun. Cycle-Time Key Factor Identification and Prediction in Semiconductor Manufacturing Using Machine Learning and Data Mining. *IEEE Transactions on Semiconductor Manufacturing*, 24(2):237–248, 2011.

[MM15] Junichi Mori and Vladimir Mahalec. Planning and scheduling of steel plates production. Part I: Estimation of production times via hybrid Bayesian networks for large domain of discrete variables. *Computers & Chemical Engineering*, 79:113–134, 2015.

[MMRS55] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, 1955. Available at: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html. Last visited on 06.01.2022.

[MNVdC22] Saulo Martiello Mastelini, Felipe Kenji Nakano, Celine Vens, and Andre Carlos Ponce Leon Ferreira de Carvalho. Online Extra Trees Regressor. *IEEE transactions on neural networks and learning systems*, PP, 2022.

[Mon13] Douglas C. Montgomery. *Design and analysis of experiments*. Wiley, Hoboken, NJ, 8. ed. edition, 2013.

[MPCOF$^+$17] Fernando Martínez-Plumed, Lidia Contreras-Ochando, Cèsar Ferri, Peter Flach, José Hernández-Orallo, Meelis Kull, Nicolas Lachiche, and María José Ramírez-Quintana. CASP-DM: Context Aware Standard Process for Data Mining, 2017.

[MRT18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts and London, England, second edition edition, 2018.

[Mur12] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts and London, England, 2012.

[Nil98] Nils J. Nilsson. *Artificial intelligence: A new synthesis*. Morgan Kaufmann, San Francisco, Calif., 1998.

[NWNW12] Hai-Long Nguyen, Yew-Kwong Woon, Wee-Keong Ng, and Li Wan. Heterogeneous Ensemble for Feature Drifts in Data Streams. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Pang-Ning Tan, Sanjay Chawla, Chin Kuan Ho, and James Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7302 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[Obj22] Object Management Group. Industry IoT Consortium, 2022. Available at: https://www.iiconsortium.org/. Last visited on 18.11.2022.

[OBUM16] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In Tobias Friedrich, Frank Neumann, and Andrew M. Sutton, editors, *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 485–492, New York, NY, USA, 07202016. ACM.

[ÖKÖ06] Atakan Öztürk, Sinan Kayalıgil, and Nur E. Özdemirel. Manufacturing lead time estimation using data mining. *European Journal of Operational Research*, 173(2):683–700, 2006.

[Olh03] Jan Olhager. Strategic positioning of the order penetration point. *International Journal of Production Economics*, 85(3):319–329, 2003.

[Pa13] Promotorengruppe Kommunikation der Forschungsunion Wirtschaft – Wissenschaft and acatech – Deutsche Akademie der Technikwissenschaften e.V. Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0: Abschlussbericht des Arbeitskreises Industrie 4.0, 2013. Available at: https://www.acatech.de/publikation/umsetzungsempfehlungen-fuer-das-zukunftsprojekt-industrie-4-0-abschlussbericht-des-arbeitskreises-industrie-4-0/. Last visited on 11.11.2022.

[Pag54] Ewan S. Page. Continuous Inspection Schemes. *Biometrika*, 41(1/2):100, 1954.

[Pet00] Johann Petrak. Fast Subsampling Performance Estimates for Classification Algorithm Selection: Technical Report TR-2000-07, 2000.

[PGKM16] András Pfeiffer, Dávid Gyulai, Botond Kádár, and László Monostori. Manufacturing Lead Time Estimation with the Combination of Simulation and Statistical Learning Methods. *Procedia CIRP*, 41:75–80, 2016.

[Pla18] Plattform Industrie 4.0. RAMI4.0 - a reference framework for digitalisation, 2018. Available at: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html. Last visited on 10.11.2022.

[Pla19] Plattform Industrie 4.0. Shaping Digital Ecosystems Globally, 2019. Available at: https://www.plattform-i40.de/IP/Redaktion/EN/Standardartikel/vision.html. Last visited on 18.11.2022.

[Pla22] Plattform Industrie 4.0. Industrie 4.0 - What is it?, 2022. Available at: https://www.plattform-i40.de/IP/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html. Last visited on 18.11.2022.

[PMG98] David L. Poole, Alan K. Mackworth, and Randy Goebel. *Computational intelligence: A logical approach*. Oxford Univ. Press, New York and Oxford, 1998.

[PVG$^+$11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu

Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[REF16] REFA Bundesverband e.V. Beschreibung von 30 Standardmethoden des Industrial Engineerings. In *Industrial Engineering*, pages 45–279. REFA Bundesverband e.V, Darmstadt, 2016.

[RHS20] Christoph Raab, Moritz Heusinger, and Frank-Michael Schleif. Reactive Soft Prototype Computing for Concept Drift Streams. *Neurocomputing*, 416:340–351, 2020.

[RN16] Stuart J. Russell and Peter Norvig. *Artificial intelligence: A modern approach*. Always learning. Pearson, Boston and Columbus and Indianapolis, third edition, global edition edition, 2016.

[Rul03] RuleQuest Research. Cubist, 2003. Available at: https://www.rulequest.com/index.html. Last visited on 21.04.2021.

[RW03] Wenny H.M. Raaymakers and Ton Weijters. Makespan estimation in batch process industries: A comparison between regression analysis and neural networks. *European Journal of Operational Research*, 145:14–30, 2003.

[Sam59] Arthur L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[Sau20] Frederick Sauermann. *Datenbasierte Prognose und Planung auftragsspezifischer Übergangszeiten*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2020.

[SB18] Richard S. Sutton and Andrew Barto. *Reinforcement learning: An introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts and London, England, second edition edition, 2018.

[Sch06] Günther Schuh. *Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte*. VDI-Buch. Springer, Berlin, 3., völlig neu bearb. aufl. edition, 2006.

[SGS$^+$20] Günther Schuh, Andreas Gützlaff, Frederick Sauermann, Oliver Kaul, and Nicolas Klein. Databased prediction and planning of order-specific transition times. *Procedia CIRP*, 93:885–890, 2020.

[SGST20] Günther Schuh, Andreas Gützlaff, Frederick Sauermann, and Theresa Theunissen. Application of time series data mining for the prediction of transition times in production. *Procedia CIRP*, 93:897–902, 2020.

[SH16] David Silver and Demis Hassabis. AlphaGo: Mastering the ancient game of Go with Machine Learning, 2016. Available at: https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html. Last visited on 04.02.2022.

[SHH$^+$18] Melissa Seitz, Lasse Härtel, Marco Hübner, Lukas Merkel, Johannes be Isa, Friederike Engehausen, Matthias Schmidhuber, Frederick Sauermann, and Philipp Hünnekes. PPS-Report 2017/2018. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 113(12):840–844, 2018.

[Shi17] Christoph Mingtao Shi. „Made in China 2025“: Chinas Vision zu Industrie 4.0. *Wirtschaftsinformatik & Management*, 9(2):70–77, 2017.

[SHPS19] Frederick Sauermann, Marcel Hagemann, Jan-Philipp Prote, and Günther Schuh. Control loop for a databased prediction of order-specific transition times. In Jens Peter Wulfsberg, Wolfgang Hintze, and Bernd-Arno Behrens, editors, *Production at the leading edge of technology*, pages 463–472. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.

[SKK$^+$23] Viktor Schubert, Steffen Kuehner, Tobias Krauss, Martin Trat, and Janek Bender. Towards a B2B integration framework for smart services in Industry 4.0. *Procedia Computer Science*, 217:1649–1659, 2023.

[SPH$^+$19] Günther Schuh, Jan-Philipp Prote, Philipp Hünnekes, Frederick Sauermann, and Lukas Stratmann. Impact of Modeling Production Knowledge for a Data Based Prediction of Transition Times. In Farhad Ameri, Kathryn E. Stecke, Gregor von Cieminski, and Dimitris Kiritsis, editors, *Advances in Production Management Systems. Production Management for* *the Factory of the Future*, volume 566 of *IFIP Advances in Information and Communication Technology*, pages 341–348. Springer International Publishing, Cham, 2019.

[SPLS18] Günther Schuh, Jan-Phillip Prote, Melanie Luckert, and Frederick Sauermann. Determination of order specific transition times for improving the adherence to delivery dates by using data mining algorithms. *Procedia CIRP*, 72:169–173, 2018.

[SPM+19] Günther Schuh, Jan-Philipp Prote, Marco Molitor, Frederick Sauermann, and Seth Schmitz. Databased learning of influencing factors in order specific transition times. *Procedia Manufacturing*, 31:356–362, 2019.

[SPSF19] Günther Schuh, Jan-Philipp Prote, Frederick Sauermann, and Bastian Franzkoch. Databased prediction of order-specific transition times. *CIRP Annals*, 68(1):467–470, 2019.

[SSA13] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-Task Bayesian Optimization. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc, 2013.

[SST16] Ashish Sabharwal, Horst Samulowitz, and Gerald Tesauro. Selecting Near-Optimal Learners via Incremental Data Allocation, 2016. Available at: https://arxiv.org/abs/1601.00024.

[SSW+16] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[STB+22] Chenwei Sun, Martin Trat, Janek Bender, Jivka Ovtcharova, George Jeppesen, and Jan Bär. Unsupervised Anomaly Detection and Root Cause Analysis for an Industrial Press Machine based on Skip-Connected Autoencoder. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 681–686. IEEE, 2022.

[SWV17] Micah J. Smith, Roy Wedge, and Kalyan Veeramachaneni. FeatureHub: Towards Collaborative Data Science. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 590–600. IEEE, 2017.

[TBO23] Martin Trat, Janek Bender, and Jivka Ovtcharova. Sensitivity-Based Optimization of Unsupervised Drift Detection for Categorical Data Streams. *KIT Scientific Working Papers ; 208*, 2023.

[The23] The pandas development team. pandas-dev/pandas: Pandas, 2023. Available at: https://pandas.pydata.org/. Last visited on 24.02.2023.

[THHLB13] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, Ramasamy Uthurusamy, Inderjit S. Dhillon, and Yehuda Koren, editors, *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, New York, NY, USA, 2013. ACM.

[TO23] Martin Trat and Jivka Ovtcharova. Designing concept drift detection ensembles: A survey. In *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2023.

[TP98] Sebastian Thrun and Lorien Pratt. Learning to Learn: Introduction and Overview. In Sebastian Thrun and Lorien Pratt, editors, *Learning to Learn*, pages 3–17. Springer US, Boston, MA, 1998.

[Tuk77] John W. Tukey. *Exploratory data analysis*. Addison-Wesley series in behavioral science Quantitative methods. Addison-Wesley, Reading, Mass., 1977.

[Van18] Joaquin Vanschoren. Meta-Learning: A Survey, 2018. Available at: http://arxiv.org/pdf/1810.03548v1.

[Vap98] Vladimir Vapnik. *Statistical learning theory*. A Wiley-Interscience publication. Wiley, New York and Weinheim, 1998.

[VDB04] Antal Van Den Bosch. Wrapped progressive sampling search for optimizing learning algorithm parameters. In *Proceedings of the Sixteenth Belgian-Dutch Conference on Artificial Intelligence*, pages 219–226, 2004.

[WFH16] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Elsevier Reference Monographs, s.l., 4. aufl. edition, 2016.

[WH00] Rüdiger Wirth and Jochen Hipp. CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 2000.

[WHC+16] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

[Wol17] Eberhard Wolff. *Microservices: Flexible software architecture*. Addison-Wesley, Boston and Munich, 2017.

[WTB+24] Hendro Wicaksono, Martin Trat, Atit Bashyal, Tina Boroukhian, Mine Felder, Mischa Ahrens, Janek Bender, Sebastian Groß, Daniel Steiner, Christoph July, Christoph Dorus, and Thorsten Zoerner. Artificial-intelligence-enabled dynamic demand response system for maximizing the use of renewable electricity in production processes. *The International Journal of Advanced Manufacturing Technology*, 2024.

[WYD+13] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. Online feature selection with streaming features. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1178–1192, 2013.

[YWDP14] Kui Yu, Xindong Wu, Wei Ding, and Jian Pei. Towards Scalable and Accurate Online Feature Selection for Big Data. In *2014 IEEE International Conference on Data Mining*, pages 660–669. IEEE, 2014.

[ZC18] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: Principles and techniques for data scientists*. O'Reilly, Beijing and Boston and Farnham and Sebastopol and Tokyo, first edition edition, April 2018.

[ZVSL18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710. IEEE, 2018.