Evaluation and feasibility of selected data-driven Machine Learning approaches for Production Planning to enhance Order Sequencing and to improve OEE in Manufacturing

Master Thesis

Master of Science in Information Systems

Nicolai Christian Frosch

September 16, 2024

Supervisor:

1st: Prof. Dr. Christian Ledig
2nd: Sebastian Gocker

Chair of Explainable Machine Learning
Faculty of Information Systems and Applied Computer Sciences
Otto-Friedrich-University Bamberg

Abstract

The manufacturing industries are under pressure to enhance their productivity and efficiency in their operations due to the rising competition and market risks. Overall Equipment Effectiveness (OEE) measures the effectiveness of the equipment in use during the production process and helps in identifying the losses that occur. Despite the fact that OEE depends on several operational factors including production scheduling, machine downtime and performance, there is a lack of research on the use of Machine Learning (ML) in predicting and enhancing OEE especially in the context of production planning and control. This thesis aims at identifying how the use of ML algorithms can improve the production scheduling in the pharmaceutical manufacturing industry. The objective is to find out whether the ML models can predict OEE and other Key Performance Indicators (KPI) and how these predictions can help the human planners to create better schedules. Together with Dr. Pfleger Arzneimittel GmbH, the project includes work in the fields of data integration, feature engineering, model training and validation, and the design of a user interface that is focused product changeover simulations and generating scheduling proposals. The findings show that the ML models can predict OEE with a certain level of accuracy and low error margin. However, subcomponents of OEE and production process times observe larger errors or lower determination coefficients. Through comparing and contrasting different experimental configurations, this thesis tests the effects of feature scaling and encoding, as well as hyperparameter optimization on the model's performance. Among the various ML algorithms used, ensemble methods, especially gradient boosting, emerged as the best performing models. Furthermore, explainability of the models was improved to company's stakeholders by using SHAP (SHapley Additive exPlanations) values. The best performing trained models per target KPI were incorporated into a web-based user interface prototype. This thesis proves that the use of ML for predicting production process KPIs is a viable method for enhancing the effectiveness of production planning. Although the development of fully autonomous scheduling is not within the scope of this research, the application of ML models as a decision support tool can greatly contribute to the minimization of operational losses and improvement of overall equipment effectiveness.

Abstract

Die verarbeitende Industrie steht unter dem Druck, ihre Produktivität und Effizienz in ihren Betrieben aufgrund des zunehmenden Wettbewerbs und der Marktrisiken zu verbessern. Die Gesamtanlageneffektivität (Overall Equipment Effectiveness, OEE) misst die Effektivität der während des Produktionsprozesses eingesetzten Anlagen und hilft bei der Ermittlung der auftretenden Verluste. Trotz der Tatsache, dass die OEE von verschiedenen betrieblichen Faktoren wie Produktionsplanung, Maschinenstillstand und Leistung abhängt, gibt es einen Mangel an Forschung über den Einsatz von maschinellem Lernen (ML) bei der Vorhersage und Verbesserung der OEE, insbesondere im Zusammenhang mit der Produktionsplanung und -steuerung. In dieser Arbeit soll untersucht werden, wie der Einsatz von ML-Algorithmen die Produktionsplanung in der pharmazeutischen Fertigungsindustrie verbessern kann. Ziel ist es, herauszufinden, ob die ML-Modelle die OEE und andere Key Performance Indicators (KPI) vorhersagen können und wie diese Vorhersagen den menschlichen Planern helfen können, bessere Pläne zu erstellen. Gemeinsam mit der Dr. Pfleger Arzneimittel GmbH umfasst das Projekt Arbeiten in den Bereichen Datenintegration, Feature-Engineering, Modelltraining und -validierung sowie den Entwurf einer Benutzeroberfläche, die auf Produktumstellungssimulationen und die Erstellung von Planungsvorschlägen ausgerichtet ist. Die Ergebnisse zeigen, dass die ML-Modelle die OEE mit einem gewissen Genauigkeitsgrad und einer geringen Fehlerspanne vorhersagen können. Allerdings weisen Teilkomponenten der OEE und der Produktionsprozesszeiten größere Fehler oder geringere Bestimmungskoeffizienten auf. Durch den Vergleich und die Gegenüberstellung verschiedener experimenteller Konfigurationen werden in dieser Arbeit die Auswirkungen der Merkmalsskalierung und -kodierung sowie der Optimierung der Hyperparameter auf die Leistung des Modells getestet. Unter den verschiedenen verwendeten ML-Algorithmen erwiesen sich Ensemble-Methoden, insbesondere Gradient Boosting, als die leistungsfähigsten Modelle. Außerdem wurde die Erklärbarkeit der Modelle für die Stakeholder des Unternehmens durch die Verwendung von SHAP-Werten (SHapley Additive exPlanations) verbessert. Die leistungsfähigsten trainierten Modelle pro Ziel-KPI wurden in einen Prototyp einer webbasierten Benutzeroberfläche integriert. Diese Arbeit beweist, dass der Einsatz von ML für die Vorhersage von Produktionsprozess-KPIs eine praktikable Methode ist, um die Effektivität der Produktionsplanung zu verbessern. Obwohl die Entwicklung einer vollständig autonomen Planung nicht in den Rahmen dieser Forschungsarbeit fällt, kann die Anwendung von ML-Modellen als Entscheidungshilfe einen großen Beitrag zur Minimierung von Betriebsverlusten und zur Verbesserung der Gesamteffizienz der Anlagen leisten.

Acknowledgements

I want to thank my supervisors, Prof. Christan Ledig and Sebastian Gocker, who were always available to support me in the project, whenever questions occurred. Furthermore, I would like to thank Christian Stemper, who started the project with his initial ideas and support, even though he left the company throughout the projects timeline.

Furthermore, many thanks to the cooperation company, Dr. Pfleger Arzneimittel GmbH, and all stakeholders in the project for the great cooperation and support throughout the timeline of the thesis project, especially to the Cloud and Datateam, who supported me in topics of data exploration, understanding and validation.

Next, I give special thanks Valentin Lange, Sebastian Falkner, Max Holz and Andre Theumer who supported me in proofreading the thesis and discussing ideas related to the thesis practical project.

Finally, I want to thank my friends, family and girlfriend for supporting me throughout the time of writing this thesis and in general my educational career.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
AR	Availability Rate
DTR	Decision Tree Regression Regression
DSR	Design Science Research
DT	Down Time
GMM	Gaussian Mixture Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
k-NNR	K-Nearest Neighbors Regression
KPI	Key Performance Indicator
LGBM	Light Gradient-Boosting Machine
LIME	Local Interpretable Model-Agnostic Explanations
LR	Linear Regression
MAE	Mean Average Error
MES	Manufacturing Executive System
ML	Machine Learning
MLP	Multi Layered Perceptron
MSE	Mean Squared Error
NN	Neural Network
OEE	Overall Equipment Effectiveness
OFE	Overall Factory Effectiveness
OT	Operating Time
PE	Performance Efficiency
PEE	Production Equipment Effectiveness
PPC	Production Planning and Control
PPT	Planned Production Time
PR	Polynomial Regression
QR	Quality Rate
R2	R^2 Coefficient of Determination
RF	Random Forest
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RR	Ridge Regression Regression
SA	Simulated Annealing
SHAP	SHapley Additive exPlanations
SL	Supervised Learning
SQL	Structured Query Language
SVR	Support Vector Regression
SVM	Support Vector Machine
TEEP	Total Equipment Effectiveness Performance
UI	User Interface
USL	Unsupervised Learning
XGB	eXtreme Gradient Boosting
xAI	Explainable Artificial Intelligence

Notation

a	A scalar (integer or real)
$\mathbf{x}$	A vector representing the input features in machine learning models
$\mathbf{w}$	A vector representing the weights in machine learning models
$\mathbf{w}^\top$	The transpose of the weight vector
$\hat{y}$	The predicted target value (output)
y	The actual target value (output)
$\boldsymbol{a}$	A vector (generic)
$\boldsymbol{x}^i$	The i-th example (input) from a dataset
y_i or $\boldsymbol{y}_i$	The target associated with $\boldsymbol{x}^i$ for supervised learning
$\|\mathbf{w}\|_2^2$	The squared L_2 norm of the weight vector
b	The bias term in linear regression and other models
n	The number of samples in the dataset
p	The number of predictors in a model
d	The degree of the polynomial in polynomial regression
ϵ	Epsilon-tube in SVR, defining the margin of tolerance
$f(\mathbf{x})$	A function mapping input vector $\mathbf{x}$ to an output
$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
σ	Standard deviation
μ	Mean
$\{0, 1\}$	The set containing 0 and 1
a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
$\|\|\boldsymbol{x}\|\|_p$	The L_p norm of vector $\boldsymbol{x}$

1 Introduction

"The current manufacturing environment is characterized by high complexity, dynamic production conditions, and volatile markets" (Usuga Cadavid et al., 2020, p. 1531). Manufacturing companies strive to optimize their productivity and profits in this competitive environment. One of the primary factors of production, and consequently one of the most significant cost shares in manufacturing companies, are the machines used in the production process. To achieve productivity increases multiple approaches are used by manufacturing companies like smart or predictive maintenance, smart production planning and scheduling, process control and monitoring, quality control, as well as smart design of products and processes (Usuga Cadavid et al., 2020).
Due to this drive for optimization, numerous Key Performance Indicator (KPI)s have been developed to measure production performance. One of the most commonly used KPIs and tools to gauge operational excellence in manufacturing is the Overall Equipment Effectiveness (OEE) (Pintelon and Muchiri, 2008). The OEE "measures different types of production losses and indicates areas of process improvement" (Pintelon and Muchiri, 2008, p. 2). Its subcomponents combine information about the quality of the production process, the performance of machine usage, and the availability or downtime of production machines into a single value ranging from zero to one hundred. This breakdown of operational excellence into a single KPI allows for easy comparisons between companies and situations as well as evaluation of analysis of potential actions to improve operational excellence. Because OEE summarizes the efficiency of machine usage and therefore has a strong effect on profitability of companies operations, manufacturing companies strive to achieve the highest possible OEE. While OEE can be influenced by operational processes like production scheduling and control, selection of appropriate machines, or raw material selection, it is also susceptible to external factors like employee engagement, worker productivity, staff sickness, experience, supply chain issues, and many more (Pintelon and Muchiri, 2008).
The present thesis explores whether OEE can be increased with the help of ML techniques. In a practical implementation project, in cooperation with Dr. Pfleger Arzneimittel GmbH, a medium-sized German pharmaceutical company, various machine learning methods will be implemented to predict several different KPIs. Dr. Pfleger aims to improve its operational excellence by improving the availability component of the OEE with the help of smart production planning and scheduling. These machine learning methods are used to predict KPI related to the production planning process and enable the simulation of product changeovers regarding their effectiveness. In practice, these predictions allow human planners always to choose the most efficient product sequence regarding a chosen KPI with the hope of reducing losses in the production process and increasing efficiency.
The project in the present thesis can be classified in the framework of Usuga Cadavid et al. (2020)'s meta-analysis as a "Smart Planning and Scheduling" use case. The project spanned about one year, with the first months revolving around data exploration and integration, gathering business knowledge, and finally conceptualizing a final tool with defined business requirements. Afterward, an implementation phase of about nine months followed by a loose structure and rapid development iterations in a prototyping style. Finally, the implementation results and trained ML-models were evaluated against the defined requirements.

The following subchapters will outline the business project motivation, explain the research methodology used, and describes the further thesis structure.

1.1 Project Motivation

Human planners often make decisions in production planning and scheduling, sometimes without a factual or data basis — in other words, by human gut feeling. This focus on human decision-making is the case at Dr. Pfleger. In a case study, this thesis aims to evaluate whether the decision-making process in production planning can be made more efficient using machine learning methods to predict the OEE. The primary motivation of Dr. Pfleger is their general goal of transforming into a data-driven company and gathering first experiences in the realm of Artificial Intelligence (AI) and ML. Additionally, creating a data basis and tooling to predict the OEE would allow for evaluating the current planning process. If a sufficiently performant solution is reached, machine learning models provided via a web-based user interface will support the humans in the product planning process. A fully autonomous production planning system based on AI and machine learning is a long-term vision of Dr. Pfleger but seems out of reach due to missing factors on OEE in the available data. Therefore, the focus of this case study and implementation project is set to the prediction of individual production orders and the calculation of scheduling proposals with the help of trained ML models. The business perspective and problem is discussed in more detail in Chapter 3.

1.2 Research methodology

This chapter outlines the research approach and methodology of this thesis. Generally, the thesis aims to answer whether the OEE can be predicted with the help of machine learning methods and evaluate which machine learning methods are suitable to the task and data at hand. Therefore, the approach and goal of the thesis are to conduct a practical implementation of a functional ML-based tool for the cooperation partner, including data processing, feature creation, model training, and evaluation, as well as creating a simple user interface to interact with the models trained. The following research questions describe the research employed in this thesis:

1. "Which machine learning algorithms are suitable for predicting the OEE?"
2. "Which loss function are suitable for training models predicting the OEE?"
3. "Is it possible to predict the OEE using ML-algorithms with the data available at Dr. Pfleger with reasonable performance?"
4. "Which of trained model and technique family performs best to predict the OEE?"
5. "Is it possible to predict further OEE related KPIs with using ML-algorithms with the data available at Dr. Pfleger with reasonable performance?"
6. "How can the prediction of the OEE be used to enhance order sequencing?"
7. "Is it possible to generate reasonable scheduling proposals for human planners with the help of the models trained?"

The thesis aims to answer the above research questions by providing a theoretical background (Questions 1 & 2) and implementing a machine learning tool for Dr. Pfleger (Questions 3-6). It should be noted that this would be an exemplary answer, especially questions 3 to 6, whose results can hardly be generalized from this thesis as it highly depends on the specific data and circumstances at the site of Dr. Pfleger. Generally, the ratio of theoretical to implementation-related research questions highlights the focus on the practical application of machine learning methods.
This thesis follows the methodology of action research. Wilde and Hess (2007) describes action research as solving a practical problem using scientific and practical methods in multiple cycles of analysis, action, and evaluation steps. Additionally, the methodology and structure of the present thesis are inspired by the technique of Design Science Research (DSR) as outlined by Holzweißig (2019) (compare Figure 1). The DSR approach is designed to provide a structured methodology when creating new software artifacts in a research context. Even though the DSR methodology is focused on classical software development, it can also be applied to this context of data science and machine learning. In the framework of DSR, this thesis starts at the research entry point "Design & Development centered initiation". The reason for this entry point is that the problem identification was done by Dr. Pfleger beforehand, and motivation, as well as the general objectives of a solution, were already defined by or together with Dr. Pfleger before the actual implementation project began. Therefore, this thesis is mainly concerned

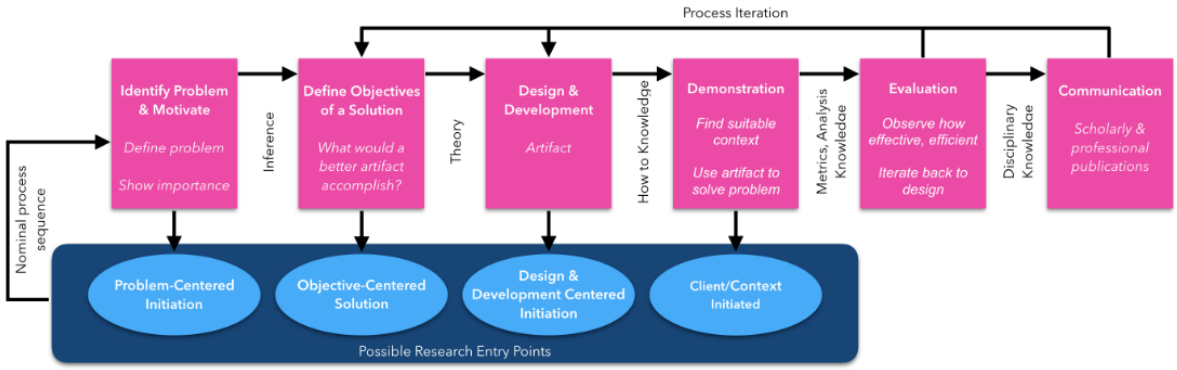


Figure 1: Design Science Research Process after Pfeffers (Holzweißig, 2019)

with the implementation conceptualization, development, and evaluation of iteratively creating a ML-based tool to predict the OEE. Over the nine months of the implementation phase, which were preceded by three months of concept discussions and work on data integration, weekly or bi-weekly meetings with the responsible from the company have been conducted. Each week, the project's current goals and status were discussed with the company's stakeholders, results were presented, feedback was given, and further implementation iterations were planned. After each finished implementation goal and evaluation of the results, design discussions for further improvements were done. The process described above resembles the general concept of the design science research methodology with a focus on short-cycle implementation. Finally, after the conclusion of the entire project, an overall evaluation against the initially proposed goals and defined business requirements was conducted together with the stakeholders at Dr. Pfleger to judge the project's outcome.

The following subchapter will briefly describe the structure of the thesis, which is based on the above-described methods by Wilde and Hess (2007) and Holzweißig (2019).

1.3 Thesis Structure

The present thesis is orienting itself after the DSR methodology proposed by Holzweißig (2019) as discussed in the previous chapter. To follow the DSR process Holzweißig (2019) provides a structure recommendation for scientific papers. The structure proposal by Holzweißig (2019) splits the main part of common thesis structures into four chapters: As-Is analysis, target concept, implementation, and evaluation. Additionally, he proposes the chapter's introduction and theory at the beginning and critical reflection and outlook at the end of the thesis. The present thesis uses a version of the structure proposals with adjustments to the recommendation to account for the data-oriented nature and the business environment in which the thesis is created.

This thesis follows the recommendation for the first two chapters. The chapter Introduction provides the motivation and introduces the research methodology and structure. The Theoretical Background provides the necessary knowledge to understand the further chapters of the thesis and introduces important concepts used in the implementation like loss function, machine learning algorithms, or how the OEE is calculated. The theoretical background is not going into technical and theoretical depth in most topics because it assumes that the reader has prior knowledge about machine learning and data-related topics as well as a basic understanding of programming and algorithms.

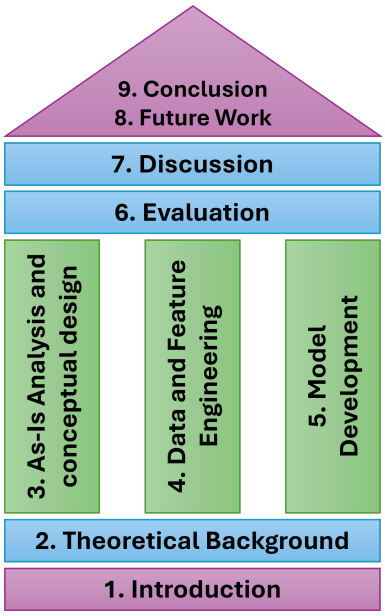


Figure 2: Structure of the paper based on Holzweißig (2019, p. 40)

The thesis deviates from the structure proposal primarily in the main part. The As-Is analysis and target concept have been combined into one chapter "As-Is Analysis and conceptual design" because the business environment had no tooling similar to the one developed in this project that could have been analyzed. Instead the business environment and production planning processes have been described. Additionally, the chapter

contains a description of the defined target concept and the formalized business requirements. The proposed implementation chapter has been split into several thematic areas, which represent different parts of the tooling or data pipeline. First, there is the chapter "Data and Feature Engineering", which describes the implementation of a pipeline for data gathering, modeling, and integration of the provided raw data. The aggregated data result of this pipeline is then used for further feature engineering steps, which are also described in this chapter. Furthermore, the fourth chapter discusses the data quality issues throughout the data integration implementation and how they have been dealt with. Finally, a description of the resulting dataset, the distribution characteristics of certain features and target values are provided. The chapter "Model Development" gives insights into the process and techniques used for model training, validation, and explanation. The chapter discusses the chosen loss functions, which dataset splits have been applied, how ML models were selected, which target values were predicted, used data encoding and scaling methods, and how hyperparameter optimization was employed. Additionally, the chapter defines certain experimental training setups that will later be evaluated. The chapter also discusses how models can be explained using ML-Explainability methods. A further part of the model development chapter is the model provision in a user interface. This part of the chapter explains how the trained models have been stored, managed, and finally provided to the company's internal users via a developed web-based user interface. Furthermore, the chapter explains how the trained models can be used to generate planning proposals with different search algorithms. In addition to the proposed chapter on critical reflection and outlook, a chapter, "Evaluation," has been added. It provides the results of the model training and proposed experimental setups as well as the evaluation of customer requirement fulfillment satisfaction. The initially proposed chapters have been slightly split and renamed as "Discussion", "Future Work", and "Conclusion". The overall thesis structure can be seen in Figure 2, inspired by a similar figure provided by Holzweißig (2019) for his structure recommendation.

2 Theoretical Background

The following chapter provides the theoretical background necessary to understand the further chapters of the thesis. Note that this chapter means to give a high-level overview of concepts used in the practical and evaluation part of this thesis, like KPI definitions, high-level explanation of used machine learning algorithms and methods, as well as loss functions employed to evaluate model performance. Therefore, the chapter creates an theoretical basis to explain technical methods used and choices made in the implementation process. This chapter is not meant to provide the reader with a fundamental understanding of how machine learning, its mathematical operations, or its various algorithms work or to explain all terms related to the technical area of data science. Therefore, the thesis expects the reader to have basic knowledge of programming and technical and mathematical concepts in machine learning.

First, this chapter will provide a list of basic definitions of frequently used terms in the further thesis. Secondly, the environment where the thesis topic revolves, Production Planning and Control (PPC), is discussed shortly. Then, the main KPI, which is to be predicted in this thesis, the OEE, its calculation, its sub metrics, and related KPIs, will be described. Furthermore, machine learning concepts and methods, such as loss functions, learning methods, and model explainability options, are briefly introduced and explained. Finally, after providing an introduction and explanation of the essential terms, chapter 2.5 highlights the research environment in which this thesis is located.

2.1 Basic definitions

A list of basic definitions is provided below to help readers understand the following chapters of the thesis, related concepts, and frequently used terms in the machine learning and data context. This list is supposed to create a shared understanding of what specific terms and abbreviations used within the paper mean. Several terms in the definition list will be discussed further in the coming subchapters.

Table 1: Key Definitions from Literature

Term	Definition
Industry 4.0	Refers to the fourth industrial revolution, integrating cyber-physical systems, IoT, and automation to create smart, real-time optimized manufacturing processes (Oztemel and Gursev, 2020, p. 127-128).
Production Planning & Control	Refers to the planning, scheduling, and monitoring activities in the production process to ensure efficient use of resources and timely delivery of products (Chapman, 2005, p. 12).
Information	Information is the usable answer to a concrete question (Zehnder, 1998, p. 14).
Redundancy	Redundancy describes multiple occurrences of the same statement (Zehnder, 1998, p. 27).
Consistency	Data is called consistent if it satisfies the consistency conditions specified for a dataset (Zehnder, 1998, p. 27).
Database	An independent, permanent organization form designed for flexible and secure use, usually consisting of the database itself and associated data management (Zehnder, 1998, p. 35).
Structured	A database is called structured when systematic subdivisions and links are possible (Zehnder, 1998, p. 26).

Term	Definition
Machine Learning	ML is a field of study where algorithms learn from data to improve performance on tasks without being explicitly programmed to solve them (Goodfellow et al., 2016, p. 98-107).
Artificial Intelligence	The broader concept of machines or systems mimicking human-like intelligence and behavior (Goodfellow et al., 2016, p. 98-107).
Neural Networks	NN are a class of machine learning models composed of multiple layers of interconnected nodes with its structure being inspired by the human brain's neural architecture (Goodfellow et al., 2016, p. 13).
Deep Learning	DL is a subset of machine learning that uses neural networks with a high number of layers to model complex patterns in data (Goodfellow et al., 2016, p. 98-107).
Training Error	Errors occurring when training a model on a training data set are called training errors (Goodfellow et al., 2016, p. 110).
Test Error (Generalization)	Errors occurring on unseen inputs from the test set are called test or validation errors and describe the models generalization capability (Goodfellow et al., 2016, p. 125).
Capacity	The capacity describes a ML-model's ability to fit to the complexity of functions, with low-capacity models underfitting and high-capacity models overfitting (Goodfellow et al., 2016, p. 111-113).
Underfitting	Underfitting describes when a model is too simple to capture the complexity in the training data resulting in high training error (Goodfellow et al., 2016, p. 112-115).
Overfitting	Overfitting describes when a model is too complex, capturing the noise in the training data, leading to a large differences between training and test errors (Goodfellow et al., 2016, p. 112-115).

2.2 Production Planning and Control

Production Planning and Control (PPC) is a core methodology used in managing production operations in both manufacturing and service industries. PPC deals with planning, scheduling, and control of production-related activities in a company to ensure that operations run smoothly, meet customer demands, and in order to maintain efficiency within the company. According to Chapman (2005) PPC is used to balance demand and resource availability, optimize usage of the available resources, and finally ensure that production activities are in line with business goals (Chapman, 2005, p. 12).

To achieve this goal, PPC uses several tools and activities like forecasting demand, creating efficient production schedules, managing inventory, and controlling the flow of materials, goods, and information throughout a company's production process. These activities are crucial for manufacturing companies to maintain their competitive edge in today's highly dynamic business environment, where customer demands, and production technologies are continuously evolving (Chapman, 2005, p. 2).

One key challenge in research about PPC is addressing the differences between service and manufacturing operations. Manufacturing companies typically deal with physical goods and products, which can be stored as inventory to buffer against fluctuations in supply or demand. In contrast, most service companies often lack these options because their products are intangible and cannot be stored. This leads to a greater focus on timing, customer involvement, and quality in service industries (Chapman, 2005, p. 3).

In terms of the process categories within production planning and control, different production environments, like make-to-stock, make-to-order, and assemble-to-order, require different planning and control approaches. Each production environment has its own challenges regarding the activities discussed before (Chapman, 2005, p. 3-4). A make-to-stock environment refers to producing goods based on assumed or forecasted demand and storing them until customer orders are placed. This allows for quicker fulfillment

but requires efficient inventory management. On the other hand, make-to-order starts production only after receiving a customer order therefore reducing inventory levels while increasing lead times. In an assemble-to-order environment, pre-manufactured components are assembled once a customer order is placed, which allows to strike the balance between flexibility in the production process and inventory management (Chapman, 2005, p. 3-5).

A medium-sized pharmaceutical manufacturing company like Dr. Pfleger can be categorized in different ways. Most appropriately it might fall under the make-to-stock category, as pharmaceutical products are typically produced in large batches based on forecasted demand to ensure availability and compliance with stringent regulatory requirements. This is only true for the company's own products and not for commissioned production. The latter might be considered into the assemble-to-order category as products are only manufactured when an order is received, with input materials being already in stock most of the time.

The above-discussed concepts and activities summarized as PPC are essential in determining a manufacturing company's financial performance and market competitiveness. Efficient scheduling of production activities ensures optimized use of a company's resources. It minimizes potentially existing production bottlenecks, which reduces losses in the process and, therefore, reduces costs and improves overall profitability. Insufficient production scheduling can result in excess production materials, product inventory, and underutilized resources or might cause delays in order fulfillment. All of these problems negatively affect a company's profitability and its market position (Chapman, 2005, p. 46). Finally, efficient production schedules also improve a company's speed and flexibility in responding to unexpected market fluctuations and changing customer demands and are therefore able to provide a competitive edge (Chapman, 2005, p. 92). This competitive edge gained by PPC is significant in industries and markets where just-in-time production is commonly applied (Chapman, 2005, p. 151).

In conclusion, understanding the various production environments within production planning and control is essential for designing effective strategies for a company's specific needs. It is vital to know how production planning and scheduling work based on the environment in which the company is situated. As we move forward to the next chapter the focus shifts to operational efficiency measured in its performance using the OEE metric. OEE plays a critical role in linking the production planning and control context with real-time performance monitoring by providing real-time and on-a-glance insights on the effectiveness of the production process. Regarding the topics of PPC, the activity of creating production schedules and enhancing the human-based creation of such schedules with the help of ML-methods will be the focus throughout the rest of the thesis.

The following section will explore the OEE metric and its calculation in detail and discuss its relevance for optimizing production processes within the PPC framework.

2.3 Overall Equipment Effectiveness

The OEE is one of the most widely used metrics to measure and track operational performance in production processes (Pintelon and Muchiri, 2008). It can be considered a tool in the context PPC to monitor operational efficiency and highlight areas of improvement. "The OEE tool is designed to identify losses that reduce the equipment effectiveness.

These losses are activities that absorb resources but create no value." (Pintelon and Muchiri, 2008, p. 6). Losses in the context of OEE refer to time lost due to different issues in the production process.

OEE is a compound KPI, which consists of the three metrics referred to as availability rate, performance efficiency, and quality rate (Pintelon and Muchiri, 2008). As Figure 3 shows, the OEE is defined as the result of the multiplication of the three sub-metrics. Per definition, OEE ranges from zero to one in the decimal space or from 0% to 100% percent. An OEE value of 100% is considered optimal operational performance with no losses, while an OEE close to 0% resembles a production process prone to high losses.

Figure 3: OEE Calculation (Industries, 2024b)

Scheduling losses refer to times lost from the total available time, usually measured against 24 hours a day, in which machines are unused due to times not assigned in the production schedule. This loss, therefore, means the machine was not planned to be used for production even though it could have been produced due to reasons like order shortages, material shortages, or similar issues. These scheduling losses are usually not considered when calculating the OEE (Pintelon and Muchiri, 2008).

$$\text{Availability Rate} = \frac{\text{Operating Time (hrs)}}{\text{Loading time (hrs)}} \times 100 \tag{1}$$

The Availability Rate (Eq. 1) measures the downtime losses in the production process depending on the specific definition. Downtime losses refer to planned or unplanned stops of the production process, which are the times the machine is meant to be producing but does not due to more significant issues in the machine's operation. Planned stops are times required to prepare and start the machines for production. Unplanned stops are related to unexpected and extended equipment failures that are not easily fixable. The availability rate is calculated from two subcomponents. First, the loading time, sometimes called Planned Production Time (PPT), refers to the time the equipment is planned to operate. Second is the operating time when the production equipment is running productively. It is calculated by subtracting the downtime losses from the loading time (Eq. 2) (Pintelon and Muchiri, 2008; Industries, 2024a; REFA, 2024).

$$\text{Operating Time} = \text{Loading Time} - \text{Downtime} \tag{2}$$

Performance efficiency (Eq. 3) shows smaller time losses in the production process. It can be described as the difference between the optimal or expected production speed and the actual production speed. Losses here include minor production speed losses, low-scale idle times (up to a few minutes), and minor machine or production process faults that can be fixed quickly. The performance efficiency is calculated from the Theoretical cycle time, the actual output, and the operating time. The Theoretical Cycle time is the ideal time required to manufacture one product under perfect conditions, assuming there are no interruptions in the production process at all. The Actual Output is just the amount of units produced during a given period. (Pintelon and Muchiri, 2008; Industries, 2024a; REFA, 2024).

$$\text{Performance} = \frac{\text{Theoretical Cycle time (hrs)} \times \text{Actual Output (Units)}}{\text{Operating Time (hrs)}} \tag{3}$$

Finally, the Quality rate (Eq. 4) describes the time loss caused by producing faulty products. It is the ratio of good produced products to total produced products. Higher rates of faulty production mean a lower quality rate of the production process, and more time is required to fulfill the necessary order amount of good quality products (Pintelon and Muchiri, 2008; Industries, 2024a; REFA, 2024).

$$\text{Quality Rate} = \frac{\text{Total Production - Defect Amount}}{\text{Total Production (Units)}} \times 100 \tag{4}$$

To summarize, the OEE is a KPI that allows companies to monitor the effectiveness of their equipment usage at a glance. Its subcomponents highlight specific areas for potential optimization of the production process and can, therefore, be very useful to understand where inefficiencies are created. As Figure 4 highlights, calculating the OEE is always done against a specific period. The basis for the calculation is the loading time, also called planned production time, which is the time after subtracting the scheduling loss from the whole 24 hours in a day. With other KPIs, it is possible to take the scheduling loss into account, meaning 24 hours a day as a basis. The losses described above are referred to as the traditional six big losses (Pintelon and Muchiri, 2008; Industries, 2024a; REFA, 2024). A breakdown of the six-big losses can be seen in Figure 5 and summarizes the previously discussed losses associated with each subcomponent of the OEE.

Figure 4: Overview of time losses measured by OEE subcomponents (Industries, 2024a)

Overall Equipment Effectiveness	Recommended Six Big Losses	Traditional Six Big Losses
Availability Loss	Unplanned Stops	Equipment Failure
	Planned Stops	Setup and Adjustments
Performance Loss	Small Stops	Idling and Minor Stops
	Slow Cycles	Reduced Speed
Quality Loss	Production Rejects	Process Defects
	Startup Rejects	Reduced Yield
OEE	Fully Productive Time	Valuable Operating Time

Figure 5: OEE Six big losses after Industries (2024c)

According to Industries (2024d), an OEE value above 85% is considered a world-class OEE. They also state that most companies, in reality, have OEE scores of about 60% or even lower than 45%. Generally, they explain that it is more important for a company to focus on improving its current score than on the absolute value (Industries, 2024d).

Lastly, it should be noted that out of the OEE, a system of further KPIs relevant to describe the internal and external factors on a company's production effectiveness developed as described by Pintelon and Muchiri (2008). In Pintelon and Muchiri (2008) classification, OEE describes the internal, operations-related influences on production losses, excluding the planned downtime. The Total Equipment Effectiveness Performance (TEEP) includes the planned downtime, and the Production Equipment Effectiveness (PEE) additionally includes commercial-related external reasons for no or low

demand for the product. Finally, the Overall Factory Effectiveness (OFE) takes internal and external business factors into account like stock control, internal/external logistic and supply problems, regulation, organizational problems, and natural causes Pintelon and Muchiri (2008).

From this classification by Pintelon and Muchiri (2008), it can be concluded that the present thesis, with its focus on the OEE, is related primarily to improving operational effectiveness with the help of machine learning. This focus means that the influence of an operational improvement might be influenced or even overshadowed by the worsening of the general economic situation, organizational issues, or numerous other problems inside and outside the company. In the next chapter, the background concepts of machine learning, related methods, and algorithms used in this thesis will be explained.

2.4 Machine Learning

"Machine learning is a branch of artificial intelligence that enables computers to learn from data and improve their performance on specific tasks without being explicitly programmed." (Kumar et al., 2023, p. vii). It can be described as teaching a computer to recognize and understand patterns in the data that even humans might not recognize and make decisions based on what it has learned from the data. The data used for machine learning can be anything from text to images to videos to more complex high-dimensional datasets. Machine learning as a research field has evolved a lot over the last two decades, and ML has become a widely used method to tackle everyday problems in almost any field of business and human activities. This development was fueled by the availability of more (cheap) computational power and the collection of ever more extensive (training) datasets (Haenlein and Kaplan, 2019; Jordan and Mitchell, 2015).

A machine learning algorithm is learning from a given experience in the form of given data to solve a specific class of task, and some performance metric measures its learning performance (Goodfellow et al., 2016, p. 99-105). Experience in this context means a dataset containing many examples of the task to solve (Goodfellow et al., 2016, p. 104-105). These algorithms are used to solve various tasks like classification, regression, clustering, translation, detection of data anomalies, and many more. Similarly, many different learning algorithms and performance measures can be imagined (Goodfellow et al., 2016, p. 99-103).

ML algorithms can be divided into three different learning methods: Supervised Learning (SL), Unsupervised Learning (USL), and Reinforcement Learning (RL) (Goodfellow et al., 2016). The most commonly used learning method is supervised learning, which provides the algorithm with labeled training and certain target data. Generally, SL and USL can be differentiated primarily due to what experience or data they use to learn, where the former, as mentioned, uses labeled training data, while the latter uses unlabeled training data. The algorithm shall then learn the patterns in the data connecting training and target data (Goodfellow et al., 2016). This thesis will use supervised learning to solve the regression task to predict the OEE.

In ML, three essential concepts are used to evaluate a model's learning performance. First is the error function, which refers to a mathematical formula used to compute the error between a model's prediction and the actual target value. It is applied to each individual prediction and usually measures the difference between prediction and actual. The loss

function refers to the aggregation of the outcome of the error function on all individuals across the entire dataset. This aggregation is commonly done by averaging the errors over all individuals, but sometimes weighting is applied. The loss function is used as the objective value for the model to minimize while training. Finally, performance measures allow for evaluating a model's overall performance after finishing training. Often, performance measures and loss functions are the same mathematical function where one is applied while training and the other after training solely for validation Goodfellow et al. (2016).

This chapter does not aim to explore all possible or available methods and measures but focuses on the subset most relevant to the regression task at hand. The next subchapter will briefly explain a regression problem or task and what models are available and commonly used to solve it.

2.4.1 Regression Problems and Models

A regression problem or task asks the algorithm to predict a numerical value given some data input (Goodfellow et al., 2016, p.101 & 107). To solve this task a learning algorithm is asked to output a function $f : \mathbb{R}^n \to \mathbb{R}$ (Goodfellow et al., 2016, p.101), which means predicting or calculating one value out of a multitude of given numeric input values, given in the form of a vector $\mathbf{x} \in \mathbb{R}^n$. Generally, learning behavior is emulated by most ML algortihms by searching for the optimal set of weights or rules to minimize the prediction error measured, which is done with the chosen error and loss function (Goodfellow et al., 2016, p. 107-109).

The learning algorithm called linear regression is the most commonly known and most straightforward way to solve a regression problem (compare chapter 2.4.3). Like many learning algorithms, linear regression learns by fitting the model's weights, sometimes also referred to as parameters, such that the loss function on the training set is minimized. As the name says, linear regression aims to find a linear function f that calculates the prediction based on a vector multiplication between the transposed weight vector and the input vector (Eq. 5). Due to this, it cannot learn non-linear relationships, which are pretty standard in real-world data. More complex models are required (Goodfellow et al., 2016, p. 109 & 110).

$$\hat{y} = \mathbf{w}^T\mathbf{x} = w_1x_1 + w_2x_2 + \cdots + w_nx_n \tag{5}$$

Polynomial Regression (compare Chapter 2.4.3) extends linear regression by being able to learn a polynomial function, which allows for modeling non-linear relationships in the original dataset. A linear regression model is considered a polynomial regression with a degree of one. The Support Vector Machine (SVM) algorithm searches the hyperplane that maximally separates the given data instances, and they use the kernel trick to handle non-linearly separable data by transforming it into a higher-dimensional space, where a linear separation is possible without explicitly computing the transformation. They can be used for Support Vector Regression (SVR) to solve regression problems (compare Chapter 2.4.5). K-Nearest Neighbors Regression (k-NNR) is another simple regression algorithm that predicts an instance's target value by averaging the outcomes of the instance's nearest neighbors in the feature space (Goodfellow et al., 2016).

Furthermore, machine learning algorithms based on decision trees can be used to solve regression problems. These algorithms can use only a single decision tree like Decision Tree Regression Regression (DTR) (compare Chapter 2.4.4) or ensembling methods utilizing a multitude of different decision tree models and combining their predictions to improve performance (compare Chapter 2.4.8). Neural Networks (compare Chapter 2.4.6), especially deep learning models, capture complex non-linear relationships by adjusting multiple layers of neurons based on the input data (Goodfellow et al., 2016).

Whichever algorithm is used to solve the regression task, the performance metric and loss function must be appropriately chosen to guide the model's training process toward the desired outcome. The next subchapter will discuss which loss functions apply to regression tasks and, more specifically, to the prediction of KPIs like the OEE. Afterward, the machine learning algorithms mentioned above will be explained in more detail.

2.4.2 Loss Functions for Regression Problems

Selecting an appropriate loss function for a learning task is critical in developing ML-models, which is especially true for regression tasks predicting KPIs due to the high sensitivity of KPIs like the OEE to even minor prediction errors. The selection of the error loss not only determines the optimization process during model training but also significantly impacts the accuracy and generalization of the model's later predictions. Among the various loss functions available, the Mean Squared Error (MSE) and its close derivation, the Root Mean Squared Error (RMSE), is one of the most commonly used in regression tasks across different domains (Bajaj, 2022; Goodfellow et al., 2016; Allen, 1971). Furthermore, the Mean Average Error (MAE) and the R^2 Coefficient of Determination (R2) are frequently used to analyze errors and performance of regression models. Each loss function has advantages and disadvantages based on its mathematical properties. It should be noted that the loss function and the performance metric are usually the same for regression tasks. This subchapter briefly evaluates the suitability of the various loss functions for models for predicting KPIs, motivating the choice of loss function for the present project.

The MAE, MSE, and RMSE are most effective when the prediction errors follow a normal distribution (Goodfellow et al., 2016). In many regression tasks, including those involved in KPI prediction, it is reasonable to assume that the errors are normal distributed. Under this assumption, they provide an accurate measure of model performance, ensuring the model is optimized to minimize the typical deviations from the actual values. This characteristic of MAE, MSE, and RMSE is well-documented in the literature, among others discussed by Patton and Timmermann (2007) in their evaluation of forecast optimality.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{7}$$

The defining characteristic of the MSE is its sensitivity to larger errors and outliers. This sensitivity exists because the MSE squares the prediction error before averaging them, which amplifies larger errors over than smaller ones, as can be seen in the Equations 6 and 7 (Bajaj, 2022; Goodfellow et al., 2016). In the equations, y_i represents the actual

value, while $\hat{y}_i$ represents the predicted value for the i-th data point. This sensitivity characteristic of MSE makes those two loss functions particularly effective in identifying and penalizing models that exhibit many more significant prediction errors or outliers, which is valuable in KPI prediction tasks where differences between target and predicted values can have substantial implications for the later decision-making process.

The main difference between the MSE and RMSE is that the latter measures the loss in the same units as the target and prediction values, while the former measures the loss in squared units (Bajaj, 2022). This difference in measurement has no impact on the model training itself. However, it is relevant for the interpretation of results and explainability of a model's performance to the customer, which is especially true in the present task because the error values of predictions for KPIs like OEE would become smaller due to the squaring operation, which is counter-intuitive for non-technical stakeholders in the project.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

In opposition to the (R)MSE, the MAE (Eq. 8) is robust against outliers due to not amplifying large errors by squaring them. The interpretation of MAE is intuitive and gives an easy understanding of how far predictions deviate from the actual target values. One key disadvantage of the MAE is that, unlike the (R)MSE, it is not differentiable and, therefore, not well optimizable. Therefore, it is not used as a primary loss function in this project but as a support metric to better understand the model performance, which is more robust towards outliers that data quality issues might cause (Bajaj, 2022).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{9}$$

$$R_a^2 = 1 - \left(\frac{(1-R^2)(n-1)}{n-p-1}\right) \tag{10}$$

The R^2 coefficient of determination (Eq. 9) is a common metric for assessing regression model performance, indicating how much of the variance in the target variable y is explained by the model. If $R^2 = 1$, the model explains all variability; if $R^2 = 0$, it explains none. A negative R^2 suggests the model performs worse than a simple mean-based prediction. However, R^2 can be misleading, as adding more predictors increases it, even if they do not improve the model. Adjusted R^2 (Eq. 10) accounts for the number of predictors, providing a better comparison between models with different numbers of variables. It penalizes the inclusion of unnecessary predictors, only increasing if a new variable improves model performance. Unlike R^2, adjusted R^2 does not automatically rise with more predictors, making it a more reliable measure, although it is always lower than R^2 and slightly harder to interpret (Bajaj, 2022).

In Table 2, the available loss functions and performance metrics discussed in this chapter are briefly summarized by their characteristics.

This chapter and the previous introduction chapters answers the second research question defined in chapter 1.2 with the theoretical foundation. The MAE was considered unsuitable to optimize models in this context because KPIs are very sensitive to even small changes. Larger errors are penalized by the (R)MSE which is considered advantageous

Table 2: Comparison of Loss and Performance Metrics for Regression Problems

Loss Function	**Sensitivity towards Outliers**	**Optimizable?**	**Intuitive Interpretation?**	**Explanation Only?**
MSE	Yes	Yes	No	No
RMSE	Yes	Yes	Yes	No
MAE	No	No	Yes	No
R^2	No	Yes	Yes	Yes
R_a^2	No	Yes	No	Yes

by the author for this use case. The RMSE is considered the most suitable loss function by the author to predict the OEE because its more intuitive interpretability for the stakeholders and developers in this use case compared to the MSE. The MSE would be similarly suitable for the sole purpose of model training and optimization as the RMSE.

Now the next chapters will describe the functioning of various ML-algorithms and techniques to solve regression problems. Together with a overview of previous research about machine learning applied in PPC, these chapters will provide an answer the first research question.

2.4.3 Linear, Polynomial and Ridge Regression

The introduction of several less complex regression algorithms in this chapter follows the work and notation of Goodfellow et al. (2016). Linear regression is one of the simplest algorithms of the supervised learning paradigm and has roots in the statistical field. Linear regression is the most basic type of regression analysis, and the goal of the model is to find a linear function of the inputs $\mathbf{x} \in \mathbb{R}^n$ for a scalar target y. Mathematically, the predicted output $\hat{y}$ is given by:

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b$$

where $\mathbf{w}$ is the weight vector and b is the bias. The model learns these parameters by minimizing the MSE between the model's predicted values and actual target values in the training dataset. When the MSE is minimized, this problem has a closed-form solution for the normal equations.

Polynomial regression models are an extension of linear regression models because they include polynomial terms for the input features and enable those models to learn non-linear relationships between the input features and the output. Some are non-linear and cannot be modeled by linear regression, proving that this polynomial regression is a higher-capacity model. A polynomial regression of degree d can be expressed as:

$$\hat{y} = \sum_{i=0}^{d} w_i x^i$$

While this polynomial transformation enhances the modeling capacity, it also comes with the added risk of overfitting. These risks are especially so when the polynomial degree is

too high. Such overfitting happens when the model is trained with excessive precision so that it captures noise in the data set rather than the pattern.

Some methods used to reduce overfitting include regularization, where one of the most popular techniques is ridge regression, also known as L2 regularization. Ridge regression introduces a penalty term to the chosen loss function, typically the MSE. This penalty $J(\mathbf{w})$ is proportional to the squared magnitude of the weights:

$$J(\mathbf{w}) = \text{MSE} + \lambda \|\mathbf{w}\|_2^2$$

where the regularization parameter λ that determines the degree of the regularization penalty; the higher λ, the model tries to achieve simpler solutions with smaller weights to avoid overfitting.

By incorporating higher-level mathematical features such as polynomials, balancing the model, and regularizations, one can perform better on unseen data depending on the model's capacity for the given problem. In the present project, all the techniques mentioned above are used, and the basic and simplest one is linear regression, which acts as a benchmark for all the other used algorithms.

2.4.4 Decision Tree Regression

Initially, the decision tree was a simple graph-based way to manually implement AI with predefined decision thresholds by humans. Decision trees today refer to non-parametric algorithms that work by repeatedly splitting the feature space of the given training dataset into ever smaller regions based on feature values. Each split results in a node potentially with its subtree or leaves. At each created node, the algorithm chooses a feature and a threshold value for this feature to split the data so that the output within each resulting subtree is as homogeneous as possible. This decision process continues until the created decision tree meets some initially defined stopping criterion. Commonly for decision trees, a maximum depth or minimum number of nodes per leaf is chosen as the stopping criterion (Goodfellow et al., 2016, p.145-146).

"Each node in the tree represents a decision or prediction based on a particular input variable and the tree branches represent the possible outcomes"(Kumar et al., 2023, p. 189). The so-called leaf nodes at the end of the tree, which are those nodes that do not have their own children nodes or subtrees, contain the predicted values for a provided input individual that falls into this region of the feature space. Those predicted values are, therefore, associated with the decisions made on the way to the leaf. In summary, the decision tree algorithm creates a piecewise constant approximation to the searched target function of a given classification or regression task (Goodfellow et al., 2016, p.145-146).

Even though decision trees can model non-linearity in data and target function, overfitting is also present in this case, especially when the tree is grown very deep. There are two ways to avoid such overfitting problems in decision trees: Pruning, which involves the removal of some branches in the decision tree after it has been developed, OR using regularization parameters to limit the development of the tree. This regularization works similarly to the regularization used for the previously discussed simple regression models. Visual representation of decision trees makes them highly interpretable, as one can trace the decisions made at each node based on the learned threshold values to understand

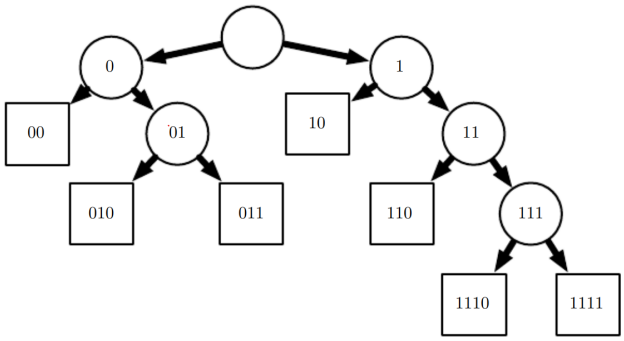


Figure 6: Decision Tree Concept (Goodfellow et al., 2016, p. 145)

the final prediction made by the model (Goodfellow et al., 2016, p.145-146). Figure 6 presents an example of a decision tree with binary decision boundaries and includes only zeros and ones at each decision step.

While trees can be as simple as this figure, they are usually significantly deeper and can have more leaves than just one per node. Despite their simplicity and interpretability, decision trees have their inbuilt limitations, including sensitivity to small changes in the data and a general tendency to create overly complex trees, which leads to higher computational costs. The latter issue can be prevented by choosing suitable hyperparameters for modern implementations of the decision tree algorithm.

Ensemble learning methods like random forests or boosting algorithms like XGBoost can resolve many weaknesses of single decision tree models by combining the predictions of multiple learned decision trees to reduce variance and improve predictive performance (Goodfellow et al., 2016, p.145-146). Such ensembling methods will be discussed shortly in the chapter 2.4.8.

2.4.5 Support Vector Regression

SVR is a modification of the SVM, which is mainly used for classification but also used for regression problems. SVR, in its essence, determines the function that differs from the actual observed values not more than ϵ and, at the same time, is as flat as possible to generalize on unseen data which is especially important in regression tasks (Goodfellow et al., 2016, p. 141-142).

SVR's structure is relatively simple. They create a decision boundary that partitions data points without regard to its class but rather by the distance to the regression hyperplane. The concept is to reduce the prediction error for data points within a margin while overlooking the errors within the ϵ-tube. The error is deemed acceptable if the predicted value falls within this range. This is because the model should not be penalized for close values of the actual values (Goodfellow et al., 2016, p. 141-142).

Another essential feature of SVR is the application of the so-called kernel trick. This trick enables the algorithm to work in high-dimensional feature space by substituting a dot product with a kernel function. This kernel trick uses the Gaussian kernel, or radial basis function since it enables the algorithm to model the non-linear relation between features and outputs. The kernel function works out the similarity of the data points in this transformed space, and thus, the model can perform optimally given highly non-linear data. The dependency of the input features on the prediction function in this

transformed space is linear, making it computationally efficient to optimize (Goodfellow et al., 2016, p. 141).

Nevertheless, SVR has a major drawback regarding computational complexity, which becomes a big issue when working with big data. The disadvantage is that the kernel function must be computed for every pair of training samples during the training and prediction processes. To this end, the model utilizes a portion of the training set referred to as the support vectors, which are the data points outside the ϵ-tube. The support vectors determine the position and direction of the regression hyperplane. As such, by only focusing on these vectors, the model can achieve efficiency while at the same time maintaining accuracy (Goodfellow et al., 2016, p. 142).

In conclusion, SVR is a very useful and versatile technique for regression problems, especially when the data is not linear, as seen with the kernel trick. Nevertheless, due to its high computational complexity, it can be slower when processing large amounts of data is required.

2.4.6 Neural Networks for Regression

Neural networks have become popular in regression tasks due to their capability of capturing non-linear relationships in the data. The explanation of the neural network's working principle and the subsequent introduction provided here are taken from Goodfellow et al. (2016) chapter six.

Simply put, a neural network is a structure with layers of nodes or neurons that are connected in some way. Every neuron takes the input data, performs a weighted sum, and then applies an activation function to it. Multi Layered Perceptron (MLP) is a popular architecture for neural networks. MLPs consists of an input layer, one or multiple hidden layers and the output layer; this makes it a basic structure for regression models.

The network is trained to change weights between neurons in a manner called backpropagation to learn from given data. Backpropagation, together with gradient descent, enables the network to adjust the weights to reduce the error between the network's output and the expected output. Every iteration of the model helps it get closer to the optimal solution, which is why neural networks are very efficient for problems with large data and complex relations between features.

A major strength of neural networks is their ability to capture non-linear relationships, which is something that linear regression models cannot do. In the real-world dataset, this flexibility enables neural networks to learn many dependencies that would have otherwise been overlooked. Furthermore, the neural networks are not limited to the shallow architecture such as MLP; the deep learning architectures, where there are many hidden layers, increase the ability of the network to learn from the data and thus increase the regression performance.

For instance, deeper networks can grasp more abstract features than shallow ones and are suitable for complex data sets. However, deeper networks may consume more computational power and are vulnerable to overfitting; however, techniques such as dropout and regularization can be used to prevent these problems, thus making neural networks suitable for regression problems.

Hence, the neural networks, through multiple layers of abstraction and non-linear transformations, provide a flexible and effective tool for solving regression problems, especially in complex relationships.

2.4.7 Early Stopping

Early stopping is a way of stopping the training process of the neural networks in order to prevent overfitting of the model especially when the model is large and complex. The idea behind early stopping is simple: during training a model is evaluated using the training set and a separate validation set. Typically, when training continues, the error rate of the training set will reduce, but the error rate of the validation set will rise at some point due to overfitting of the training set (Goodfellow et al., 2016, p. 246-247).

Early stopping works by checking the model's performance on the validation set in each iteration or epoch and stopping the model training when the performance on the validation set does not continue to increase further. This method enables the model to stop when it has picked up enough information to predict new data without being overly complex and memorizing the training data. This technique benefits neural networks with numerous layers or parameters because overfitting is a frequent problem (Goodfellow et al., 2016, p. 246-247).

Thus, early stopping serves as a preventative measure for stopping the model training at the right time to learn the essential patterns of the data without overfitting and thus enhance the model's performance.

2.4.8 Ensemble Learning

Ensemble learning is a fundamental machine learning method that involves using multiple models to make predictions to improve the overall performance of the models. Ensemble methods are different from traditional learning algorithms as they combine the predictions of several algorithms to give more accurate and reliable results. Otherwise, several models of the same algorithm with different initializations, training sets, or parameters can be used. The concept is that the different models used in the ensembling process will have different errors; hence, the overall error will be reduced, and the predictions will be accurate.

Two of the most common techniques of creating an ensemble are called bagging and boosting. Bagging is a method that can be described as Bootstrap Aggregating, which means that several models are created via training on different subsets of the overall training data set, which can be obtained by bootstrapping. The predicted outputs of each trained model are then combined in the following manner: the predicted outputs are averaged in regression problems, and the predicted outputs are voted in classification problems. Ensembling is especially useful in minimizing the variance in the predictions made and avoiding overfitting. An example of bagging is the Random Forest, where several decision trees are used to make a prediction to increase the model's accuracy (Goodfellow et al., 2016, p. 256-258).

On the other hand, boosting ensembling operates in a different way. It operates on a concept of training models in a sequential manner where each subsequent model is trained to correct the mistakes of the preceding model. Boosting techniques help decrease

the model's bias, which enables the ensemble to correct its mistakes in successive steps. AdaBoost is one of the most famous boosting algorithms, which uses the strategy of adjusting the weights of the misclassified instances so that they will gain more focus in the subsequent rounds. However, overfitting is a common problem that occurs if the boosting is not appropriately controlled (Goodfellow et al., 2016, p. 256-258).

Ensemble learning can be beneficial in production planning and predicting KPIs. In this way, the ensemble methods can improve the accuracy and decrease the error of the KPI predictions, and thus, the decisions made on their basis will be more justified. The bagging is used to stabilize the predictions, and the boosting is used to fine-tune the predictions, which is more efficient than the individual models when it comes to real-world data with complex and non-linear relationships.

2.4.9 Data Encoding and Standardization

In machine learning, it is crucial to deal with categorical features since these features should be presented in a numerical form suitable for most machine learning algorithms. This is especially true for regression tasks because these features can have several types of representation, for example, binary, nominal, or ordinal. Binary data is a form of binary decision, that is, a decision between two options; these options can be yes or no but can also be any two options, for instance, day or night. Nominal data are categorical data that have no order. Ordinal data is characterized by the fact that it has a certain order of categories inherent in it. For example, color (red, blue, green) is a nominal feature while education level (high school, bachelor's, master's) is ordinal.

Most machine learning algorithms work with numerical data, so categorical features should be converted into numbers. Some machine learning algorithms are specifically designed to work with categorical variables or have inbuilt encoding functions. This thesis mainly employs algorithms that take numerical data as input without any other information. Some standard encoding methods are label encoding, ordinal encoding, and One Hot Encoding. Categorical features are often converted into the values of zero and one and, therefore, do not need special encoding methods. Of course, they can also be encoded with any other methods if desired.

Label encoding assigns a new integer to each category of the given nominal or ordinal feature without consideration for their actual order. For instance, in a color feature, 'red' may be assigned the code 1, 'blue' the code 2, and 'green' the code 3. Ordinal encoding is ideal for data with a particular order, as the name suggests. For example, speed levels can be represented with 'slow' as 1, 'moderate' as 2, and 'fast' as 3. The assignment of the concrete value can be necessary for the real-world representation of the machine learning model as, in most cases, the algorithms consider two instances as more different if the numerical representation is quite different.

One-hot encoding generates a binary column for every category in a feature. For instance, if the color feature has 'red', 'blue', and 'green', then three binary columns will be generated, and each row will have 1 in the column corresponding to the data point's color. One hot encoding is a very effective data representation method since it can represent any data and cause large feature spaces when dealing with categorical features with many categories. This could affect the rate of convergence of the model, the memory used by the model, or the efficacy of the produced model.

After data is numerically encoded, it is usually standardized or scaled, particularly when working with algorithms sensitive to the range of input variables, such as SVM or neural networks. The most popular method is Standard Scaling, which ensures that all the features have a mean of zero and a standard deviation of one. The formula for standard scaling is: The formula for standard scaling is:

$$z = \frac{x - \mu}{\sigma}$$

where x is the feature value, μ is the mean of the feature and σ is the standard deviation of the feature. Using the formula for each example of feature the whole dataset can be scaled to eliminate the differences in scaling of all features. Thus, by normalizing the data, we can ensure that every feature will be equally important for the model's learning process.

Thus, data encoding and standardization are crucial preprocessing steps that help machine learning models work with categorical data and enhance the results for datasets where feature scaling is crucial for models like SVM and NN.

2.4.10 k-Fold Cross Validation

The popular k-fold cross-validation strategy is in machine learning for training and testing models. Cross-validation is a process that is based on k subgroups, which are derived from the original data set and are referred to as "folds". The cross-validation algorithm iterates k times over the entire dataset. In each iteration k-1 folds will be used as trainig sets, while the remaining fold can be used for validation of the model. Furthermore, model's performance after training with k-fold cross-validation is calculated by averaging the results of all the k folds to get a more precise and reliable assessment of the model's performance and its capability of generalization (Fadheli, 2024). The other reason why k-

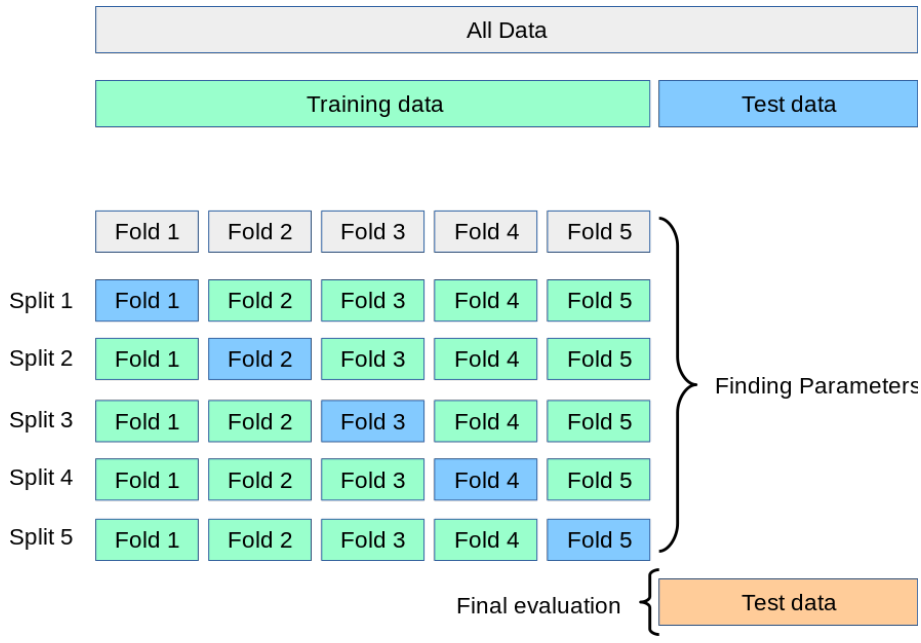


Figure 7: K-fold cross-validation concept (Fadheli, 2024)

fold cross-validation is essential is because it helps make the best use of the data, especially

when the data set is small. In contrast, the hold-out method can be very unstable due to the somewhat ad hoc way the data is split into training and validation sets. The k-fold cross-validation helps mitigate this problem by ensuring that each data point is used for training and validation (Jung, 2018). Furthermore, this technique can give a better idea about the model's performance on the unseen data, especially when applied to classification or regression problems Wong and Yang (2017). K fold cross-validation is one of the simplest and most effective methods of cross-validation of the model and, in many cases, eliminates extra computational cost and is therefore used extensively for model training in the present project.

2.4.11 Hyperparameter Optimization Methods

One of the interesting tasks in machine learning is hyperparameter tuning, which is indispensable to achieving the highest accuracy of the model. Hyperparameters are a form of adjustment mechanism within models that help identify the best results with unseen data. There are several hyperparameter tuning techniques, such as grid search, random search, and more advanced search techniques like Optuna.

The basic grid search method is the exhaustive or permutational search, where all possible combinations of the set hyperparameters are tested. The technique goes through every hyperparameter defined in the search space and trains the model in each possible hyperparameter combination. Although grid search is easy to apply and guarantees the selection of the best combination of parameters within the search space, the amount of calculations grows fast with the number of hyperparameters. This inefficiency makes grid search less usable, especially for problems or models that have numerous hyperparameters and usually occur on a large scale (Bergstra and Bengio, 2012).

An alternative to grid search is the so-called random search. It is less time-consuming because it randomly chooses hyperparameter values and their combinations for assessment. Compared to grid search, random search may not necessarily search the entire solution space but may concomitantly offer a higher-performing approximation in less time. It has been revealed in the experiment of Bergstra and Bengio (2012) that random search is usually more effective than grid Search, mainly when only a few hyperparameters define model performance. Generally, the random search method helps search large solution spaces and is more performant than a grid Search, though it depends on the model and implementation (Bergstra and Bengio, 2012).

Optuna is a second-generation hyperparameter optimization method that utilizes Bayesian optimization and makes efficient pruning to improve hyperparameters. While Grid Search and Random Search provide a fixed search space, Optuna adaptively constructs the space by using the "define-by-run" type of API to construct search space as trials go on. Those that do not perform well are pruned by Optuna's mechanism early, which is a good way of preserving computational costs. Due to the second property, Optuna is particularly effective in large-scale models, for example, neural networks, in which training is costly. In some of them, Optuna surpasses other methods, such as Random Search and Grid Search, particularly in intricate procedures like neural architecture search and hyperparameters optimization for deep learning models (Akiba et al., 2019).

In summary, though Grid Search helps thoroughly search hyperparameter space, it is much too demanding regarding computational costs. Random search is useful because

it works by picking combinations of hyperparameters at random, decreasing computation costs and usually still finding very good parameter values. However, Optuna is a next-generation approach that dynamically optimizes hyperparameters based on Bayes optimization and uses efficient pruning strategies. Therefore, it works with larger and more complex machine-learning tasks. For the task at hand all methods have been tested and due to performance limitations only random search and optuna were applicable to the practical problem in this thesis.

2.4.12 Model Explainability with SHAP-Values

The term explainability in the context of machine learning refers to making decisions made by a model understandable to human stakeholders. It is particularly important in domains where human trust, transparency, and accountability are critical, like healthcare, finance, or in the case of this thesis production planning. The latter requires human trust as production planning is one of the most critical business decisions for manufacturing companies with high influence on financial success. Without providing explainability, models often act as "black boxes," where it is difficult to know for a human why a certain prediction was made leading to distrust in the models abilities and its practical usage (Merrick and Taly, 2020).

There are numerous techniques within the field of, so called, Explainable Artificial Intelligence (xAI) aimed at making models more interpretable. One of these xAI techniques, which is widely used and also applied in this thesis is SHapley Additive exPlanations (SHAP). SHAP values provide a way to break down a model's predictions into different contributions values calculated from each feature. This technique is based on the mathematical theory of cooperative game theory and more specifically the Shapley value which is a method of fairly dividing the payoff of any cooperative game (Merrick and Taly, 2020).

SHAP values work by assigning each feature a specific value in a given prediction. This calculation of the contribution involves retraining the model with and without a given features, hence determining the extent of contribution of each feature in the prediction. This process enables one to determine the extent of the feature in reducing the model's prediction or increasing it or the overall impact of the feature on the average prediction. The computation of SHAP values may be computationally expensive since it entails averaging over all possible feature interactions in a given data set. However, this high complexity guarantees game theoretical fairness and thus makes SHAP values a reliable method for feature explanation. In practice, SHAP values can be visualized through several plots that help to understand the contribution of each feature for an individual prediction or across the entire model, which is both local and global explanations (Merrick and Taly, 2020).

In the context of production planning and this present project, SHAP values can assist in the enhancement of human confidence in the trained models through the understanding of the learning process of the model and the effects it has on the outcome as well as the directions of the chosen features. Therefore, SHAP values are a useful tool to explain the decision-making process of a model and its features to the users, thus increasing the model's interpretability.

2.5 Machine Learning in Production Planning and Control

As discussed briefly in the Introduction and in more detail in the further chapters, this thesis evaluates whether machine learning can help improve the OEE at the site of Dr. Pfleger as a cooperation partner. The OEE as an operational metric can primarily be influenced by production planning and control (PPC) measures (compare chapter 2.2 & 2.3). Attempts to improve PPC through the application of ML methods have been widely studied in research, as shown by the meta-analysis by (Usuga Cadavid et al., 2020).

The research into ML applied topics in the area of PPC has been done extensively, exploring a variety of use cases and using a multitude of ML techniques, each offering different benefits and drawdowns depending on the goal and context of the case study. Some used machine learning methods in production planning and control settings include neural networks (NN), decision trees, reinforcement learning (RL), and ensemble methods like random forests according to Usuga Cadavid et al. (2020). These methods have been used in this context because they can effectively manage the complexity and variability inherent in modern production environments and their related datasets, characterized by high uncertainty, machine breakdowns, and dynamic and rapid changes in demand. Today, many companies try to optimize their OEE with various approaches like predictive maintenance, smart production scheduling, smart product or process design (Usuga Cadavid et al., 2020; Zubair et al., 2021; Chikwendu et al., 2020; El Mazgualdi et al., 2021).

As Figure 8 shows that the use case of the plurality of papers related to ML in PPC focused on "Smart Planning and scheduling" according to Usuga Cadavid et al. (2020). In addition to the classification of the use cases in the above categories, Usuga Cadavid et al. (2020) determined the learning type and application of ML of the analyzed papers. The result is that the vast majority of studies use supervised learning methods like in the order of occurrence neural networks, decision trees, regression, ensemble learning, SVM, kNN, among others (Figure 9 & 10). These are the same methods discussed in the previous chapters and selected for evaluating methods in this thesis. Furthermore, each use case has been annotated with different characteristics by Usuga Cadavid et al. (2020), which determines how the ML models are applied. These characteristics are derived from the classical theory of Industry 4.0 and include self-organization of resources, self-regulation and self-learning of the production process, knowledge discovery and generation, as well as smart human interaction.

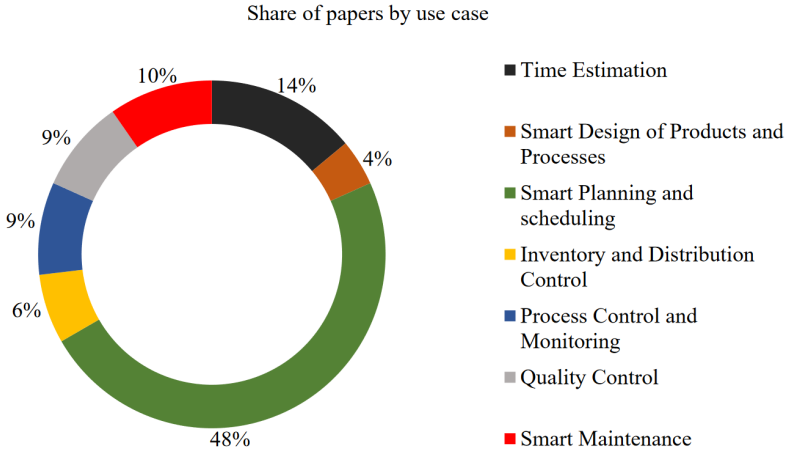


Figure 8: Share of the analyzed sample by proposed use case (Usuga Cadavid et al., 2020, p. 17)

As Figure 11 describes, the majority of papers could satisfy the characteristics connected to autonomous machine learning systems that do self-learning, self-regulation, and self-execution of self-organization within the production process. Only a minority of cases are characterized as customer, human, or environment-centric. The present paper can satisfy the characteristic smart human interaction primarily. Secondly, the characteristics of knowledge discovery and generation and self-learning of the production process are satisfied.

Usuga Cadavid et al. (2020) also provides a cross-analysis between the use case and the industry 4.0 characteristics of the paper and recognizes that most papers related to smart planning and scheduling are related to self-organization of resources, self-regulation and self-learning of the production process. Only a small minority of papers on the use case satisfied the characteristic of smart human interaction as this thesis does. Therefore, it fills a gap in this research area. Furthermore, this analysis highlights that the vision of Dr. Pfleger, which is an autonomous production planning process, aligns with the general direction of research in this area.

Finally, like the present paper, Usuga Cadavid et al. (2020) recognizes that artificial and management data dominate the research landscape in this field and disregards other data sources either due to limited availability, data quality issues, or difficulty in collecting or using further data sources. Like in this present thesis, the second most commonly used data source is product data, which describes the characteristics of the products produced Usuga Cadavid et al. (2020); El Mazgualdi et al. (2021).

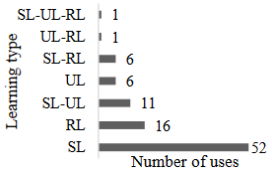


Figure 9: Number of uses by learning type (Usuga Cadavid et al., 2020, p. 14)

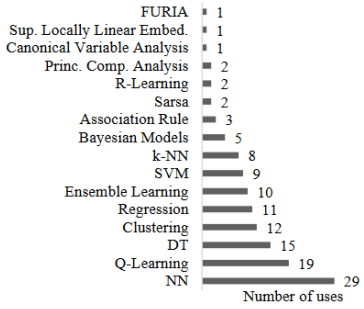


Figure 10: Number of uses by technique family (Usuga Cadavid et al., 2020, p. 13)

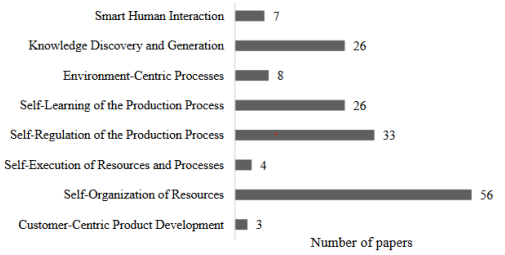


Figure 11: Number of papers by I4.0 characteristic (Usuga Cadavid et al., 2020, p. 18)

It can be concluded that the present thesis resides in the same area of research as analyzed by Usuga Cadavid et al. (2020) but focuses on one of the rarer characteristics of industry 4.0 researched so far. The next chapter will outline the business environment, define the business problem and requirements based on the analysis provided, and discuss the characteristics of machine learning in production planning and control. Furthermore, this chapter and the previous chapters briefly explained several machine learning algorithms, which are considered the answer to the first research question outlined in chapter 1.2. It should be noted that this is not an entirely conclusive answer and that several missing algorithms could likely be used and be suitable to predict the OEE. The author aimed to select a variety of algorithms from different technique families that are suitable for the task at hand in order to provide a base for evaluation. This base should provide an understanding of which technique family is suitable by example algorithms in each family.

3 As-Is Analysis and conceptual design

Now that the previous chapter provided the theoretical basis for the thesis, this chapter marks the start of the structure outlined in chapter 1.3 based on the DSR framework (Holzweißig, 2019). This exploratory prototype project is developing an entirely new tool with no predecessor. Therefore, only the human-driven process with the MES and SAP system can be described in an As-Is analysis. Additionally, this chapter will describe the project goals, the solution concept by Dr. Pfleger and requirement definition process.

3.1 Business Environment

The cooperation partner for the present thesis is a medium-sized company, Dr. Pfleger Arzneimittel GmbH, focused on the manufacturing of pharmaceutical products and located in Germany. It manufactures both its original and commissioned products and sells them primarily to pharmacies within Germany and Austria or to distributors all over the world (Pfleger, 2024b). The company provides over 60 different pharmaceutical products (Pfleger, 2024a), of which there are three general categories of products: Pills in blisters, (lozenges) tablets in tins, and ointment in tubes. Each product category is associated with different production lines in the company's production hall. All production lines consist of a clean room containing the actual production machine and a packaging machine outside the clean room that puts the actual pills, tables, or ointments into their respective packaging.

Figure 12: Pharmaceutical production and packaging machines

All production lines, furthermore referred to as lines, packaging sections are connected to a central Manufacturing Executive System (MES). The cleanroom production area is not connected to the central MES. The MES semi-automatically collects data about the packaging process after human action starts the process. This data collection is done via sensors and time trackers built into the packaging machines. Because the clean room production machines are not connected to the MES, they do not produce easily available data for further analysis. Due to this limited data availability, the project focuses only on the packaging area of the lines.

The packaging of products is mostly automatically done by the machines. The blister lines' process will be described as an example. First, the machine packs individual blisters coming out of the clean room section of the production line on an assembly line into a folding box. Then, a leaflet containing information about the medicine is added into the

folding box. Afterward, the final folding box containing the blisters and leaflet is weighed to check if it contains the right amount of individual pills or tablets to ensure product quality. If there is a deviation of more than a few grams, the product is not considered for sale but is scrapped. Now, a set of folding packages is wrapped in foil packaging into larger chunks. Finally, these product chunks are packed into a larger shipping box. These boxes are then stored and used for shipping and sales.

Even though the production lines run automatically, before production can be started, human staff needs to adjust and potentially clean the machines before a new product is produced. The effort caused by these adjustments and the cleaning process depends highly on the type of product changeover. For similar products, only minimal adjustments and no cleaning are necessary, usually taking only a few hours.

Table 3: Planned Setup Times for Different Product Changes

Type of Product Changeover	**Description**	**Assumed Worktime**
Ingredient Change	Full Line Cleaning and adjustment required	15 hours
Cartoner & Film Change	Packaging size and wrapping foil adjustments	6 hours
Film Change	Wrapping foil changes, machine adjustments	4 hours
Cartoner Change	Packaging size changes, machine adjustments	4 hours
Batch Change	Everything the same, minor adjustments	2 hours

For very different products, a complete cleaning and readjustment process of the entire line is required, which can take an entire workday or even longer if there are staff shortages. The company considers five types of product changeovers caused by different characteristics of the products. The different types of product changeovers are described in Table 3. Due to the potentially vastly different efforts depending on the time of product change, choosing a production order and minimizing these efforts becomes essential to optimize machine usage, resource efficiency, and therefore, the financial returns for the company.

3.1.1 Order Planning Process

Several human planners manage the operations of the company's production lines. Their main task is planning the order in which specific products are manufactured, taking various factors into account like material, staff, and line availability, required delivery times, and optimal order of production to minimize effort and time losses. Additionally, they need to decide which production line to use for each specific order. Some lines can only manufacture certain products. For example, there are three blister lines where each can produce roughly the same set of products. The tube production line is the only one that can manufacture the ointments in tubes though and the canning line is the only one that can produce certain pills stored in tins. Therefore, line decision-making is primarily relevant for products that can be produced on several lines, usually blister-based products.

The production schedule is planned long term, about half a year up to a year in advance. Orders are added to the schedule based on an internal sales plan for the company's products or an external manufacturing request to deliver certain products. Human planners structure the production schedule into bigger sets of orders, producing similar products and having similar delivery dates, which are so-called production campaigns. One campaign can contain five up to roughly twenty orders, but there is no rigid definition of a campaign's size. Regularly short-term adjustments to the plan must be made due to shifting delivery expectancy, material or staff shortage, or new short-notice external production requests. These can affect the production process a few weeks, one week, or even just one day ahead of the workday. To modify the production schedule, the planners

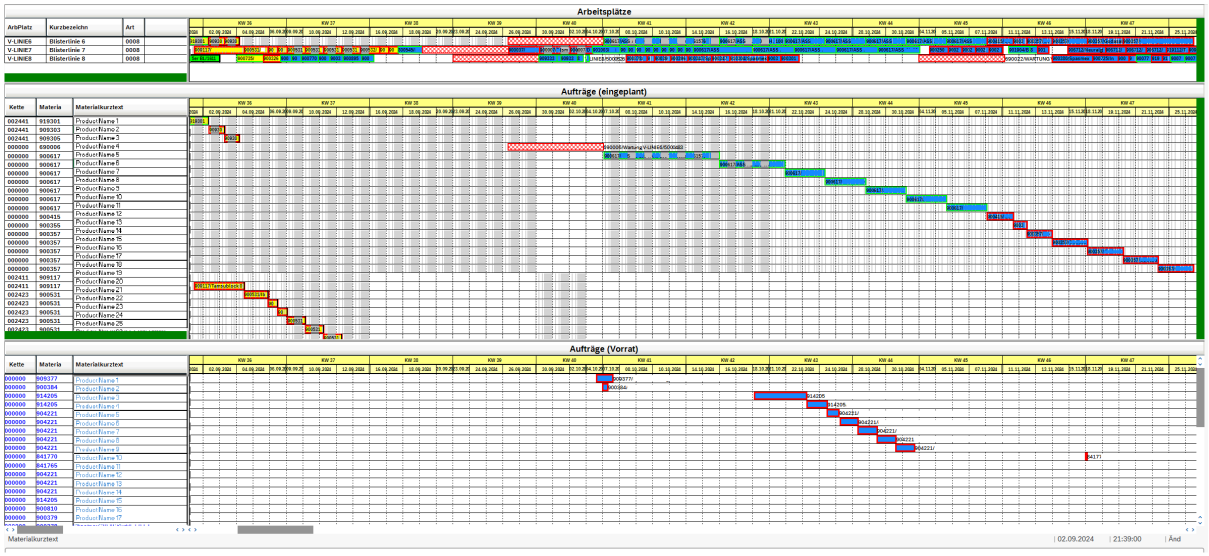


Figure 13: Production Schedule Planning Board in SAP at Dr. Pfleger

rely on an SAP-based interface. An example of this interface can be seen with redacted product names in Figure 13 for the three blister production lines. The tool displays all currently scheduled orders in a waterfall diagram, where the x-axis represents the production timeline. The length of each order in the diagram is calculated using a predefined time assumption for each product, the order quantity, and the assumed machine set-up time. The interface is split into three views: the top part line timeline view, which shows the current planning per line; the middle planned orders view, showing all scheduled orders in the waterfall diagram; and finally, the bottom order backlog view, showing all orders to be scheduled. The order backlog is the collection from which the planners must schedule each order into the overall timeline in the most time-efficient way possible. Short-term production schedule adjustments become costly because they usually involve putting products with different characteristics and small orders in short order. Therefore, short-term planning would benefit from more efficient planning and data-driven decision support.

Primarily, the work of human planners relies on human knowledge and experience rather than data-based decisions. Therefore decisionmaking behind the production scheduling follows trivially known rules, like scheduling the same products one after another to minimize losses or to prioritize larger orders, which tend to have a lower ratio of unproductive to productive time. Several simple analyses have been conducted to determine specific characteristics of the production process, like which type of product changeover requires

what amount of time. However, none of these analyses has been connected or combined into a single source of knowledge or decision support for the planners. The present project is part of a bigger drive to transform into a data-driven company and aims to provide a ML and data-based tooling to support planners in their daily work.

3.1.2 Business Problem Definition

An efficient production schedule is vital for efficient machine and staff usage to maximize the productivity and profitability of a manufacturing company. Several types of losses can occur in the production process at Dr. Pfleger, usually measured in lost time: Machine setup time losses, machine cleaning times, production volume losses due to more scrapped units, or performance losses caused by machines producing slower than expected. All these losses can be summarized in the KPI OEE and cause direct financial losses to the company, which the company wants to minimize (compare chapter 2.3).

The company has identified three areas of optimization to reduce losses in the production process: The order production sequence, the machine selection for production, and finally, the material selection for production. Due to high data availability and previous experience, optimizing the production schedule was chosen as a goal for this project. It is supposed to ground the decisionmaking of human planners in existing data by supporting and validating their decisions with the help of machine learning models capable of predicting the OEE.

Primarily, such a ML-based optimization tool would be relevant for the previously mentioned short-term adjustments to the schedule or the production sequence within one campaign because the long-term plan is usually structured in big chunks of the same or very similar products, which should be the optimal order. However, within a shorter timeframe or due to disturbances in the process causing a need for rescheduling lies the primary optimization potential. Such disturbances happen frequently at Dr. Pfleger due to staff shortages caused by sickness or vacations, material shortages, or changes in delivery dates.

The underlying assumption behind the above described goals is that a production schedule more efficient than the human plan is both possible and resulting in an increased OEE, which would symbolize a reduction in financial losses. This assumption shall be analysed and answered with the help of this thesis.

3.1.3 Technical Project Setup

Technically, the project is executed within Microsoft's Azure environment using the Azure Machine Learning Studio for the development of a data integration and model training pipeline. The company had no previous experience with the environment because its efforts previously focused on data integration and analytics tasks, as well as providing and describing data to company internal users and pharmacy customers. The data engineers of Dr. Pfleger created a connection between their Azure Storage Delta Lake, internally called "Data Factory", and the underlying database behind the MES system of the production lines. Additionally, they provided some data about product characteristics from the company's SAP system in the same Azure Storage. Data is generally provided and stored as parquet delta files, which can be loaded from Azure Storage using native environment methods.

Azure ML-Studio has extensive capabilities to train, register, and deploy ML-models and register particular predefined data storage and sources. It allows the creation of pipelines for automatic model retraining and validation. Most of these advanced capabilities were not used within the project. All code development was done with the help of native Python and standard packages like Pandas, Numpy, and Scikit-Learn. The Pytorch package was also used to develop and train a simple neural network. Furthermore, packages for more complex machine learning models that do not exist in the Scikit-Learn learn package have been used. The project focused on developing an initial codebase as an object-oriented Python package that allows for data integration and model training in the company's context. Since the primary method of execution in Azure ML-Studio is Jypyther notebooks, these have been developed as well in order to install, initialize, and run the methods within the programmed Python package. The author chose the object-oriented approach because it was unclear in which environment the models would be potentially deployed in the future. Therefore, a method executable not just within the original Azure ML Studio environment was desired. The implemented Python package is available in the thesis' Appendix.

3.2 Conceptualization of the ML-based OEE prediction tool

Now that the previous chapters have explained the situation, business process, and the business problem to be solved, the coming subchapters will describe the process of creating a concept for a ML-based tool to solve the problem. First, the process of the concept creation together with Dr. Pfleger is described and why this was the chosen path of action. Afterward, the official requirements for the projects will be defined and provided based on the framework for software system requirements by Braun (2016). These requirements are briefly evaluated in the evaluation chapter with a customer survey based on the requirements snow cards fields proposed by Braun (2016) satisfaction and dissatisfaction.

3.2.1 Project Goals and Solution Concept

The project's primary goal is to find a way to increase the overall OEE score at Dr. Pfleger. As discussed in Chapter 2.3, that means to either improve the Availability Rate (AR), the Performance Efficiency (PE), or the Quality Rate (QR). In Chapter 4.5, the data distributions of the OEE components are shown. These distributions conclude that the primary lever needing improvement at Dr. Pfleger is the AR. Because the availability rate is primarily influenced by production planning and control methods, the stakeholders at Dr. Pfleger decided to try to optimize the production planning or schedule with a ML-based tool. In summary, the project's goal in the present thesis is to develop a tool that allows Dr. Pfleger to base order planning decision-making on historical data.

The initial idea proposed by Dr. Pfleger to solve the problem was something they called the "Digital Twin" of the production lines. Throughout the initial discussions, it became clear that they described a tool capable of simulating product changeovers on their production lines by predicting their core reporting KPI in manufacturing, the OEE. An initial concept was provided by Stemper et al. (2022) including potential inputs and outputs of such a tool (compare Figure 15). Such a tool wants to predict one or multiple continuous variables, which, as discussed in Chapter 2.4.1, is considered a regression problem. In Figure 14, Stemper et al. (2022) outlines the long-term vision for such a tool of

creating an automated production and personal planning system for the company. This project focuses primarily on the first step outlined in the figure, forecasting individual orders. Additionally, methods to generate first planning proposals have been explored and are described in the chapter 5.8. After further talks with the company's project stakeholders and the company's production planners, it became clear that an initial proof of concept should first be developed as a decision support system for the human planners, helping the company to evaluate the quality of the present planning decision-making and potentially optimizing it to improve the OEE.

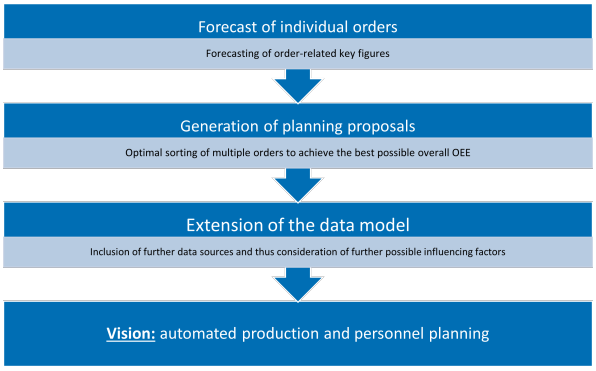


Figure 14: Long Term Vision of Dr. Pfleger (Stemper et al., 2022)

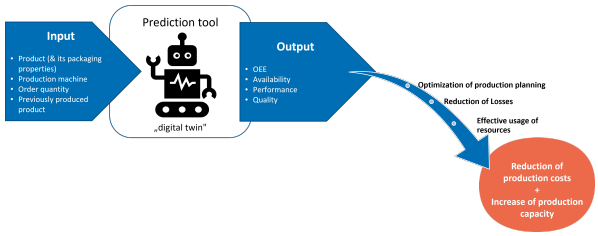


Figure 15: Solution Idea of Dr. Pfleger (Stemper et al., 2022)

Due to the nature of a decision-support tool, the explainability of models is crucial, as discussed in Chapter 2.4.12. To provide model explainability, stakeholders decided not only to try to predict the OEE itself but also its subcomponents. Additional explainability was provided by analyzing model decision-making with the help of ML explainability methods like SHAP-Values. Finally, a finished proof of concept was to have a maximum average error margin in predicting the OEE of 10% to be considered successful. This high error marring target is because the present data for the project does not include information on all relevant influences on the OEE as discussed in previous chapters.

3.2.2 Requirements Creation

In order to guide and evaluate a software development process, defined requirements are commonly used. This section briefly describes the requirement creation and evaluation process throughout the project's timeline. The project started with several discussions and meetings to determine what a tool for solving the business problem should look like. Throughout talks, vague documents and notes with requirements were taken. Finally,

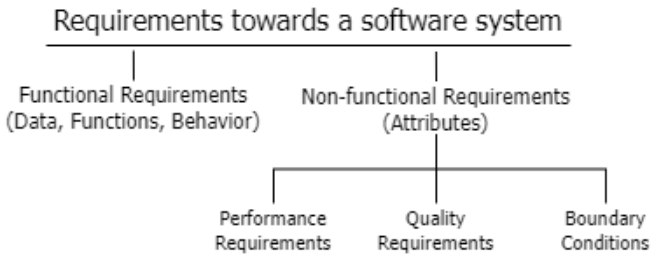


Figure 16: Requirements towards a Software System after Braun (2016)

more concrete concepts were created by the author and the company's stakeholders after the first three months of data exploration and requirements discussions. Generally,

no rigid requirements setup was used in this project. Requirements have been created, reformulated, evaluated, and started from scratch for each prototyping iteration. Each week, a review status meeting with the involved stakeholders was done, evaluating the progress in certain areas of the pipeline and discussing and adjusting requirements where needed. Table 4 provides a final list of the requirements in a short format. A description of each requirement is provided in the Appendix. The company's stakeholders did not formally define the requirements list at the beginning of the project. It was derived and formulated from the results of dozens of talks throughout the one-year implementation timespan. It follows the requirement structure given by Braun (2016) and the division between functional and non-functional requirements towards a software system (compare Figure 16). In this project's context, stakeholders defined primary requirements as vital for a functional proof of concept tool. These primary requirements have been discussed and defined in the beginning stages of the project. Throughout the development phase, further ideas for improvement and requirements emerged, including topics in the realm of ML-Operations and Deployment. These topics have been outside the project scope and could not be fulfilled. Nonetheless, for completeness, they were formulated in the table as a requirement for the desired software package. Each requirement was assigned a priority and connected to a part of one area of the data pipeline. Stakeholders have not

Table 4: List of Project Requirements

Number	Short Name	Area of Pipeline	Type	Primary	Priority
1	Data Exploration and Business Knowledge Collection		Non-functional	Yes	1
2	Implementation of Database Connection	Data Integration	Functional	Yes	1
3	Data Integration and Aggregation of MES Raw Data	Data Integration	Functional	Yes	1
4	Processing of static product and packaging information	Data Integration	Functional	Yes	2
5	Data Validation and Quality Checks	Data Integration	Functional	Yes	1
6	Feature Engineering for Model Training	Data Integration	Functional	No	2
7	Training of ML Models to predict the OEE	Model Training	Functional	Yes	1
8	Training of ML Models to predict additional KPIs	Model Training	Functional	No	3
9	Hyperparameter Tuning and Model Optimization	Model Training	Functional	No	3
10	Model Versioning and Experiment Tracking	Model Training	Non-functional	No	2
11	Model Evaluation and Selection of Best Approach	Model Evaluation	Functional	Yes	1
12	Evaluate Method for Explainability of Model Decisions	Model Evaluation	Functional	Yes	3
13	Average OEE Prediction error of below 10 percentage points	Model Evaluation	Non-functional	Yes	1
14	Implementation of User Interface Prototype for Planner Interaction	Frontend	Functional	No	4
15	Implementation of methods to allow planners to get optimized production sequences	Frontend	Functional	No	5
16	Real-World Performance Evaluation in Production Planning	Model Evaluation	Non-functional	No	2
17	Integration with Existing Production Planning Tools	Frontend	Non-functional	No	5
18	Scalability and Performance Optimization of Pipeline	Data Integration	Non-functional	No	5
19	Continuous Monitoring and Feedback Loop	Model Evaluation	Functional	No	5

explicitly assigned those priorities. Therefore, they have been approximated throughout the requirements talks and regular evaluation meetings and are subjective. Requirement one in the table is an anomaly against the usual requirement structure as it is more a requirement towards the project itself or the developer than towards the software system. It shows the primary task in the project's first week and has been added to the table for the sake of completeness. Braun (2016) describe using Snowcards with each requirement characteristic for defining and evaluating requirements. Snowcards can be a valuable tool in software development but have been unsuited for this project's fast-paced prototyping development cycles. Nevertheless, the requirements are evaluated against one element provided by the Snowcard definition: Customer Satisfaction and dissatisfaction. The author requested feedback on the satisfaction and dissatisfaction scores from the company's stakeholders at the end of the project. The evaluation chapter will analyze the overall satisfaction with the customer project based on the requirement fulfillment

4 Data and Feature Engineering

As described in the previous chapter, the core dataset for analyzing and predicting the OEE relies on semi-automatically recorded data from the company's MES. The MES is used to monitor the packaging process of pharmaceutical products across different production lines. Additionally, MES external data sources have been used and added to the dataset manually to provide more information about the products and package characteristics. These characteristics have been used for logic-based assumptions of predefined machine adjustment times, which depend on product changeover characteristics. As discussed before, the MES data was provided as delta parquet files, stored in Azure Storage, while external data was manually created by the company's responsible as Excel worksheets. Since the latter are prone to errors when humans adjust their content, they were only updated and reviewed once at the beginning of the project. In contrast, the MES data was regularly updated with new data from the live production process. The raw dataset of the MES contains data from the installation date of the system in late 2017 until now, representing roughly 4000 orders across the different production lines. It should be noted that each production line technically records its data individually, but the resulting data is stored in a common database with a common data model. This shared storage allows data analysis across the entire production facility, including all production lines. Therefore, all tables discussed in the next section contain data across all production lines, where the production line is filterable via a connected dimensional table.

The next subchapters will outline the concrete process of aggregating and preparing these datasets into a dataset usable for model training. Additionally, the steps taken for feature creation and selection will be explained. Finally, the resulting training dataset will be described and analyzed for feature balancing and characteristics.

4.1 Data Modelling

The MES data is provided as a raw data table set containing very low-level information. By the producer of the MES system, a predefined data model and associated Structured Query Language (SQL) statements for the KPI aggregation were provided to make the data usable. The data model consists of three contexts represented by their respective fact and dimensional tables (compare Figure 17). All raw data tables are connectable via predefined id fields, which were transformed into hashkeys for performance reasons in the data load step.

The first and central context within the model is the MES booking table. Each row in this table represents a work shift for the production staff when combined with the dimensional shift table. Each row can represent a morning, afternoon, or night shift and has a defined start and end time. The production counter is the second context of the data model. The counter of the packaging machine records each item it processes by using sensor detection and weighing each product. The products are then automatically classified as "good" or "scrap" to determine the QR based on their dimensions and weight. In the final dataset we end up with one row counting all products per classification category per shift. Scrapped products are items produced with quality that are considered insufficient for customer sales. Finally, the third context is the production state data, which refers to the state in which a machine is at a given time. There are various states a machine can have.

"Order change" state records the times revolving around machine cleaning and adjusting to prepare the machine for the next production phase. In the primary or secondary error state minor problems with the machine halt the production process. Lastly, the state production, which collects all times the machine is productively working. The states are changing semi-automatically, sometimes requiring specific actions from the production staff. For example, primary and secondary error states are usually automatically recorded, changed, and subsequently resolved by humans, afterward automatically changing the state again. Meanwhile, order change times get started and finished only with human input.

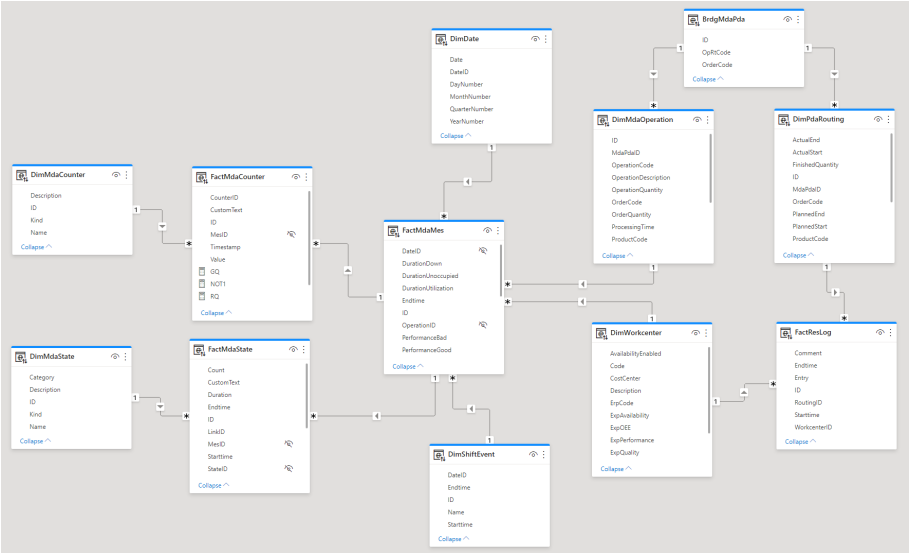


Figure 17: MES Raw Data Tables in Model Viewer

As seen in Figure 17, the context state and counter are connected via a 1:N relationship towards the MES context, which describes that shifts can be connected to various counter elements or can contain multiple production states. However, each counter or state event can only be related to a single shift. Each shift is connected to a specified operation, which is another name for a production order. Each operation contains information about the order, the product to be produced, which quantity shall be produced, and the expected processing time. Additionally, each shift is connected to a work center, a technical term for the production line or machine. Each production line has specific settings about which KPIs are tracked and what data is recorded regarding the different components of the OEE. Only lines that enabled the OEE and its sub-metrics tracking have been used for further analysis and model training. For Dr. Pfleger, all productive lines have enabled the tracking of all of the OEEs subcomponents, and only test lines are not automatically tracking KPIs.

To solve the business problem and fulfill the requirements, that were derived in third chapter, an aggregation of the three data contexts is necessary in order to calculate the KPIs on the order level and use the result as training datasets for machine learning models. This aggregation is needed because the company wants to use ML to simulate production changeovers by predicting the OEE and its subcomponents for the next order, depending on the previous one. The steps taken to perform this aggregation are described in the following subchapter.

4.2 Data Aggregation and Preparation

The previously discussed data model can transform the MES data into a usable feature set. Due to the nature of the data model differing from usual star schema models, which have only one fact table and multiple dimensional tables, more complex joins and combinations of tables are required. The MES supplier company provided the necessary data transformation steps in the form of SQL statements. Therefore, the original idea behind the aggregation logic was provided by the producer of the MES system and is not a result of this project. These SQL statements have been replicated using the Pandas package in the Python context in which the project was developed. Nonetheless, for further understanding in the coming chapters, the functionality of the data integration steps will be briefly explained.

First, an aggregation table for both the counter and the state context is generated, which is done by joining the respective fact table with the MES fact table via the MesHashKey. Afterward, the dimensional tables related to the respective context, the operation, and the work center dimensional tables are joined into the previous result via their respective identifying columns. Now that the raw data tables have been combined into a single table, we can perform aggregations with a group by statement on a unifying key. Here,

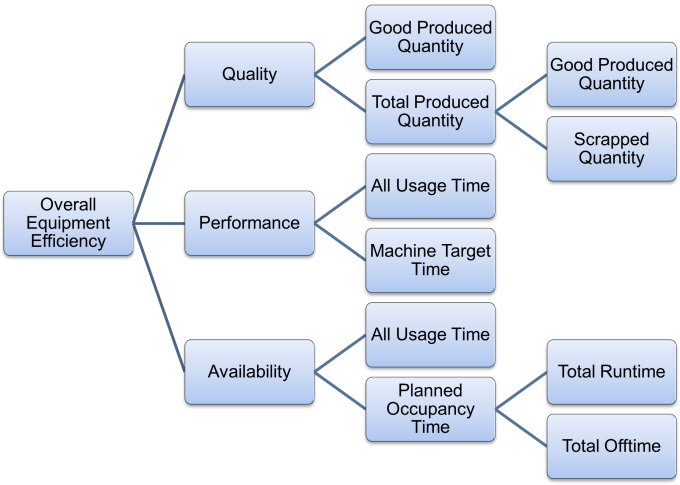


Figure 18: OEE Submetrics Hierarchy Structure

the MesHashKey is used for all aggregations because it is the central key combining all contexts. This central key is relevant primarily due to combining the results of these lower-level aggregations into a single aggregated table later. Please read the Appendix table for more details on the KPIs and their calculation logic.

After the first aggregation is done for the counter and state contexts, both result tables can be joined via the MesHashKey, and the remaining KPIs are calculated on the basis of the unique order code. As figure 18 shows, each aggregation step from right to left calculates certain sub-metrics of the OEE, where the counter context is related to the quality metric and the production state context is related to the performance, availability, and their respective sub-metrics. Vivid data quality tests have been required to validate the results of each transformation step because each step further along in the hierarchy requires correct results from the previous levels for the final result to be correct. The following subchapter will briefly detail the data quality issues and the exclusion of data caused by it.

After aggregating the MES data, two external data sources are added, which contain more information about the specific product's packaging, like package dimension, amount of blisters per package, what the active ingredient is, and similar things. Because this data is not available in the MES or the SAP systems of the company, they have been provided as Excel files and joined into the aggregated dataset via the material number, known as product code in the MES system. The data quality of these product characteristics has been validated, but because humans manually create the source, it might be prone to error. The input Excel files were stored at the beginning of the project and validated once, and in the further implementation steps, the same version was used to ensure consistency between the results of different model training.

4.3 Data Quality Issues

The quality of the training data has a significant impact on the potential outcome of training ML models. The cooperation partner decided that in-depth data validation was necessary because the models to be trained in this project should be used in a critical business process. The primary method to validate the data aggregation step was to compare its result against exporting KPIs directly from the MES's user interface. When attempting to validate the aggregated data against reports visible in and manual exports from the MES system, it became apparent that the aggregation steps only based on the provided SQL code had many orders differing in the KPI values. Initially, roughly every fifth order had differing values in at least one of the calculated metrics. Therefore, employing manual data exploration and analysis of the differing order, the root causes for the differences had to be found and, if possible, solved.

First, the SQL code provided did different calculation steps in multiple cases than the manual export. Usually, the difference was explainable by small error handlings or removal of faulty data before the data export to Excel happened, which was not explained in the provided SQL code. For example, any negative KPI values can be set to zero for export. The code in the data integration module was adjusted to account for all errors found, which were deemed related to such background functionalities of the MES front end.

Secondly, issues with the raw data have been found and handled differently by the front end of the MES tool than by the database from which the data load was executed. For example, the database still contained many rows in the fact tables of the data model regarding no longer existing or unused production lines. However, the respective dimension tables were missing some dimensional data related to these lines, which led to many errors in calculating KPIs of orders related to these production lines. The data about these lines could not be recovered and had no relevance for future prediction; because the lines were no longer existing, it was decided to remove all related entries from the dataset. About 22 of 70 thousand MES shift entries had to be omitted from further processing, significantly reducing the potential size of the training data.

Additionally, some rows or characteristics that were manually deleted from dimensional tables caused logic issues within the SQL scripts. By omitting such faulty data rows, many issues in the data quality were resolved. Furthermore, the raw dataset of the productive MES system contained several orders, either just test orders created to test the functionality of the MES or orders created but never executed in production due to cancellation or rescheduling. These orders usually resulted in an OEE value of zero

Table 5: Differing Orders in Percent of Total Dataset

Precision Threshold	**0.01**	**0.001**	**0.0001**
Differing Orders (%)	5.4%	9.3%	12.1%

or sometimes even faulty calculation, leading to negative values in one of the KPIs. Therefore, all data related to these tests or unused orders was removed from the result dataset. Finally, precision problems were caused by loading and transforming the data

Table 6: KPI Difference in Percentage Points Against Manual Validation Export

	Median			**Total**			**Maximum**		
Precision Threshold	**0.01**	**0.001**	**0.0001**	**0.01**	**0.001**	**0.0001**	**0.01**	**0.001**	**0.0001**
OEE	0.07	0.008	0.0073	19.54	20.12	20.13	2.51	2.51	2.51
Availability	0.08	0.009	0.0086	19.26	19.91	19.92	0.99	0.99	0.99
Performance	0.00	0.000	0.0000	164.24	164.26	164.27	117.30	117.30	117.30
Quality	0.00	0.000	0.0000	8.06	8.06	8.06	1.00	1.00	1.00

from the original database over the Azure storage to the final machine learning Python environment. It is unclear what caused these precision problems. The assumption is that the databases have different field precision when storing the state timestamps. These precision problems caused minuscule deviations under the fourth decimal digit in the KPIs of some orders. Because such high precision was deemed irrelevant for the further prediction or learning process, the company's responsible decided to evaluate the data quality tests with a precision threshold of 0.01. As can be seen in Table 6 and Table 5, even with all measures to remedy the differences, at the precision threshold of 0.01, 5,4% of the total dataset had differences in at least on of the KPIs, usually the availability. It was decided to omit all differing orders from the final training dataset due to a potential adverse influence on the model's training behavior.

The following subchapter briefly explains how the features were created and selected for the model training. The data integration steps partially describe and analyze the final feature set's characteristics.

4.4 Feature Creation

Most of the data aggregation steps described in the previous chapters should be seen as feature creation steps, as they strongly reduce the complexity of the original raw data by aggregating it to fit the business problem. Because this logic was inbuilt in the provided MES and necessary even to obtain the OEE values, which shall be predicted in the project, the aggregation result is considered as baseline data and not as feature creation. In addition to the predefined data aggregation steps to obtain the OEE related KPIs, further features were created based on the raw data. Primarily, these are related to the characteristics of the order itself or the product change.

Crucially, for each order in the MES dataset, the previous order was calculated based on the order's start times. This calculation was done by taking all orders in the dataset, obtaining the minimum of their shift's start times, and sorting the list by the minimum

start times for each row using the order code in the previous row as the previous order code. Naturally, this can not create a result for all orders that have no previous order at the beginning of the timeframe.

Next, a boolean flag was calculated to determine whether or not an order is a maintenance order. The MES contains entries for orders that do not refer to production processes but are only internal orders for machine maintenance. These are irrelevant to the machine learning task at hand and should, therefore, either be filtered out or ignored by the model. The flag was calculated using simple logic based on the product code associated with the maintenance orders. The product code of the maintenance orders always starts with the number eight and is one digit longer than regular product codes.

As discussed in section 2.3, the machine setup and adjustment times in the MES, referred to as order change times, significantly impact the availability rate. Similarly, the primary and secondary error times greatly impact the performance efficiency. However, these times would not be known at the time of prediction for the model. Therefore, it was decided to create features from historical times, assuming that the future characteristics and distribution of the times would be similar to those of the past. To do this, features on the dimensional level of a product changeover, which is defined as the change of one product to the next one of the next order on one production line, were calculated. Two approaches have been used: A simple mathematical or statistical approach and an approach based on unsupervised machine learning clustering the product changeover by their respective times.

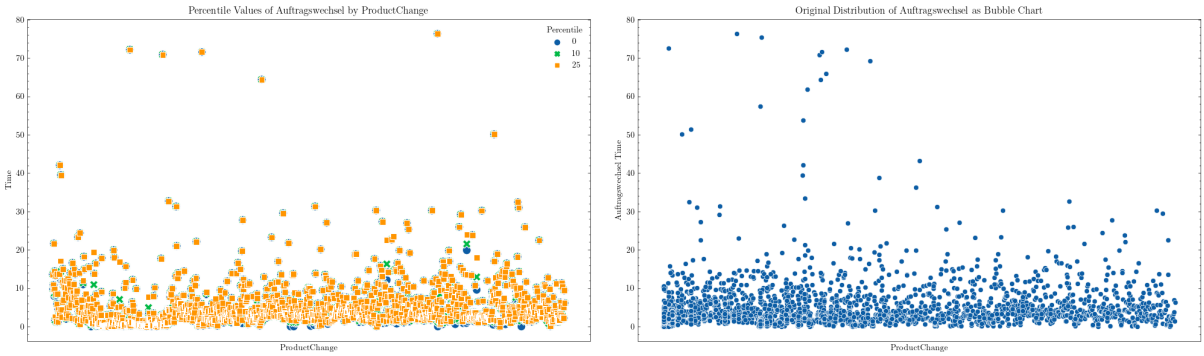


Figure 19: Product Changeover Times - Percentiles and Original Distribution

Figure 19 shows the statistical approach that calculates various percentiles of each time value per distinct product changeover and stores them as features. Here, the minimum (zero percentile), the 10th, and the 25th percentile were used. These percentiles have been chosen because the company assumed and decided that the determining characteristic of a product changeover lies in the minimum time boundary rather than potential outliers of very long times. This characteristic exists because the changeover involves different human work, which always requires the same amount of time, but sometimes is executed very slowly due to missing staff, organizational problems, or other similar problems. In addition to the percentile features, time-based clusters have been calculated with various unsupervised learning approaches like k-Nearest-Neighbor, Gaussian Mixture Model (GMM), DBSCAN, and similar models. This cluster creation was inspired by a manual analysis done by the production staff to categorize certain product changeovers into complete machine adjustment, partial adjustment, and no adjustment. After testing various methods, the GMM was chosen due to its high overlap with the manual human

evaluation time boundaries. An example result of the clustering of the GMM can be seen in Figure 20.

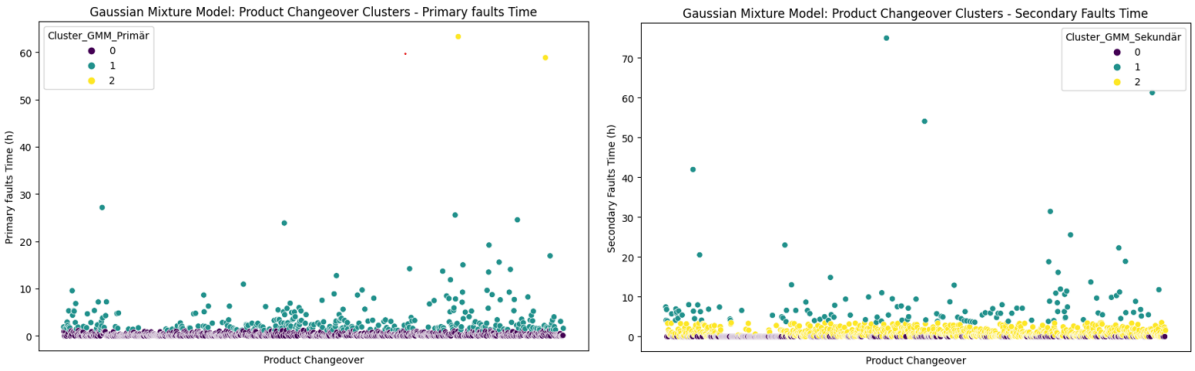


Figure 20: GMM Product Changeover - Primary and Secondary Time Clusters

Finally, features based on the external data sources have been calculated. Primarily, this was a transformation of the representation of features. The packaging dimensions in the Excel files were stored as strings, including height, width, and sometimes depth. One example of the dimensions of a product folding box is as follows: ”75x40x105mm”. Initially, this string was directly used in a label-encoded format for the model input, which resulted in poor model performance. A better approach was to split all dimensional strings into various numeric feature columns like folding box width, height, and depth, which creates a higher dimensional representation of this future. Besides the folding box dimensions, there are the dimensions of the package leaflet, which only consist of width and height. For tube products, the additional dimensions are the tube diameter and height. All created features are added based on the order code to the initially aggregated dataset and form the final integrated feature dataset together.

4.5 Dataset Description

The final integrated dataset contained 38 columnn with 4170 historical orders as rows. The intergated dataset contained historical orders from the end of 2017 until the end of July 2024 and six different production lines. Seventeen columns are considered feature columns and seven are target value columns. The target values were the OEE, its subcomponents as well as related times for planning. Another set of columns in the aggregated dataset are columns, which were just necessary in the calculation of the OEE and the other target values, but not usable as features, because they would be unknown at the time of prediction. As discussed before orders certain ones have been removed from the dataset due to either being a test order, a maintenance order or having an OEE value being considered invalid. Figure 21 shows the filtering logic in the order it is applied. The filtered dataset used for model development contained 3777 historical orders. The impact of the filtering step can be seen visually by comparing Figure 22 and Figure 23. Both figures show the distribution of the OEE and its components in histograms with 100 buckets, where the x-axis is the bucket range against the KPI

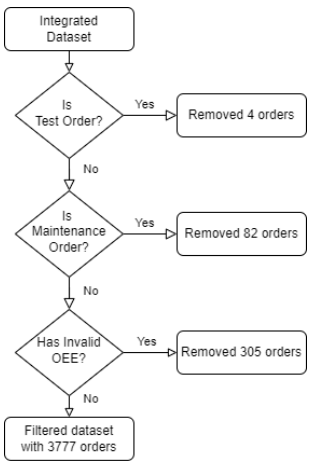


Figure 21: Dataset Filtering Logic

and the y-axis is the order amount per bucket. The data quality issues are visible in all charts pre-filtering by either having a significant bucket of zero values or, in the case of performance and OEE, a value range significantly larger than the defined 0 to 1.25 range. The filtering removes the high value ranges for performance and OEE, but still, some orders with an OEE or Performance Efficiency higher than one are present in the dataset, which means that that the performance of the machines has been faster than the defined machine speed. Usually, these faster speeds result from misdefined expected machine speeds, but they can also happen naturally in the production process. Most orders at Dr. Pfleger still have a performance efficiency below or close to one. The distributions show visually that the different components of the OEE have quite different characteristics in their mean and variance. The descriptive statistical values can be seen in Table 7.

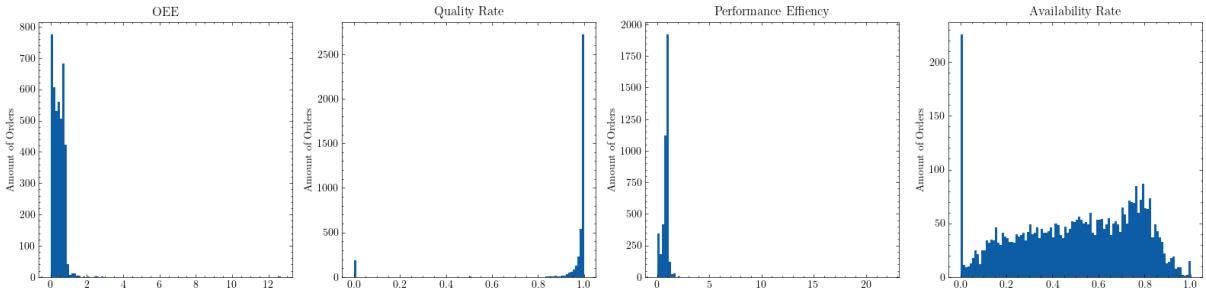


Figure 22: Dr. Pfleger - OEE and Components - Distribution pre Filter

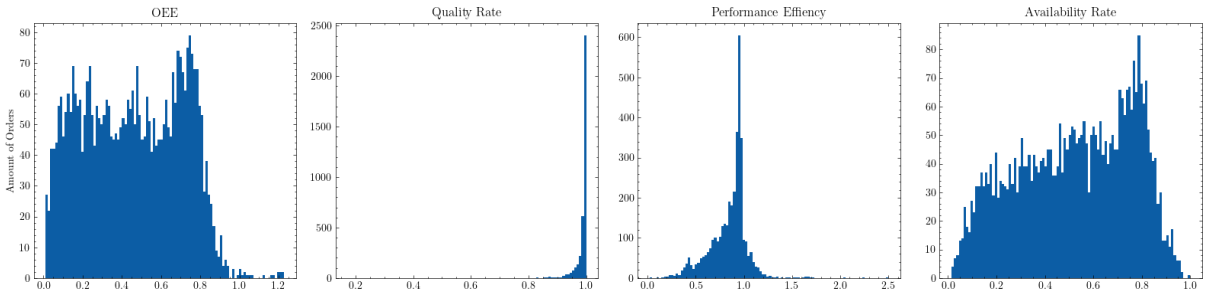


Figure 23: Dr. Pfleger - OEE and Components - OEE Distribution post Filter

OEE and the AR have similarly high variance and standard deviation. In comparison, the performance observes less variance, and the quality shows close to no variance. The latter is because Dr. Pfleger's operations and production process are optimized to obtain a near-perfect quality rate, while the primary operational problem lies in the mediocre availability rate. Due to these different distributions in the target values, it can be expected that models for the quality will be able to make more accurate predictions than models for availability rate or the OEE. Figure 24 highlights the distribution of

Table 7: Descriptive Statistics for OEE and Components

	Mean	Variance	Standard Deviation
OEE	0.457	0.062	0.249
AR	0.524	0.056	0.236
PE	0.847	0.039	0.197
QR	0.987	0.001	0.032

the model target values, which are time KPIs subcomponents of the OEE. All times seem to be distributed with characteristics close to a normal distribution. Figure 25

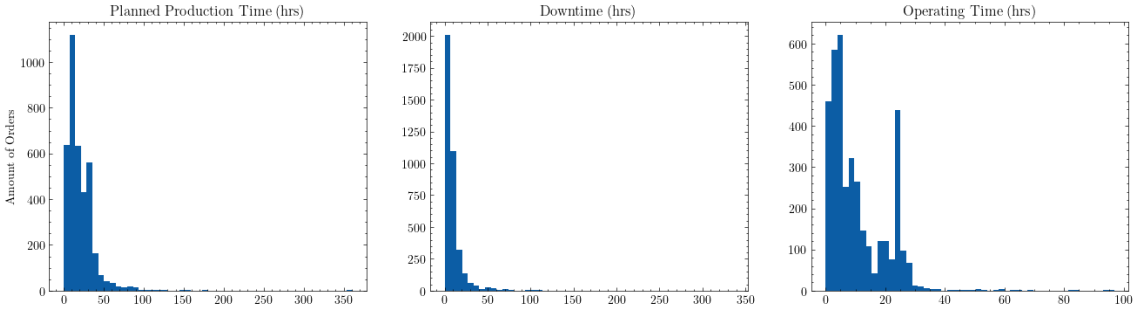


Figure 24: Dr. Pfleger - Target Values - Time Distributions

shows the characteristics and features related to the order or operation. It shows that for the blister lines "V-Linie6", "V-Linie7" and "V-Linie8" the latter tends to process significantly fewer orders. The average order quantity on the blister lines ranges between 20 and 40 thousand units. The order quantity distribution highlights that Dr. Pfleger primarily has orders with a size between 10 and 60 thousand units. There are a smaller amount of orders with larger average volumes on the lines that produce the company's own products, "V-PAST-2" and "P-SARO-2", while lines primarily used for external order production like the blister lines and the tube line tend to have smaller average order sizes. The small volume of data for the tube production line might be problematic for

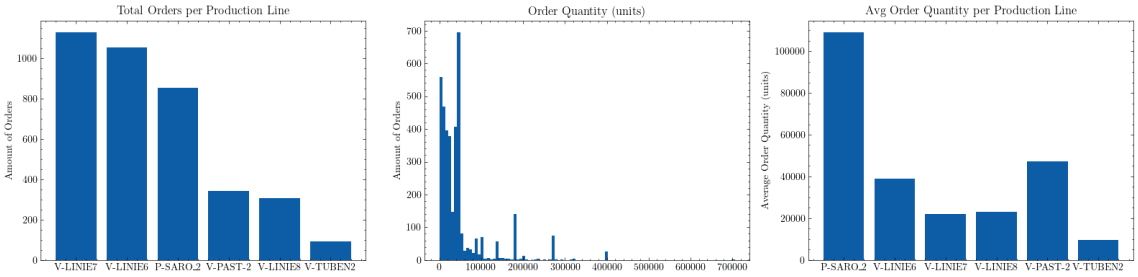


Figure 25: Dr. Pfleger - Orders per Production Line Distribution

later model training. Generally, it might be concluded from different usage characteristics in the lines that training line-specific models might yield better performance results. In the present project, only models across the entire set of production lines are trained due to the requirement to support human planners within the line selection process. Finally, Figure 26 shows the distribution of products sorted by the amout of associated

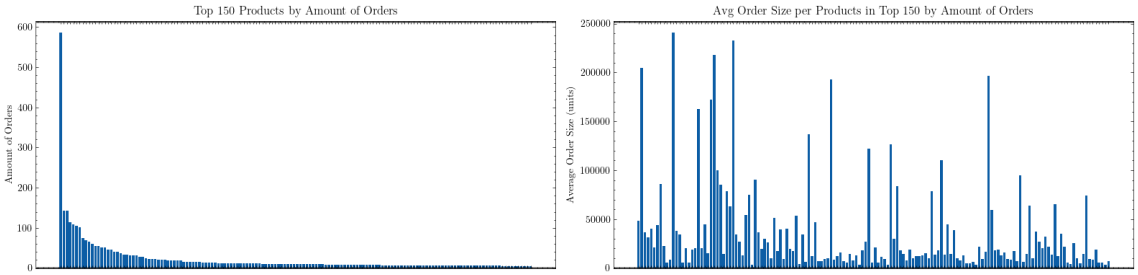


Figure 26: Dr. Pfleger - Orders per Product Distribution

orders. The figure shows only the top 150 products by the amount of orders, but in total there are 299 distinct products in the filtered dataset. Due to disclosure agreements the individual product names cannot be shown in this paper, but it is still clear from the figure that the production frequency and average order sizes vary greatly between the different products. Some orders are produced very frequently but with very small average order volumes, others are produced rarely but with high unit volumes and rarely products are produced frequently with high average order volumes.

After exploring and discussing some of the characteristics of the final dataset for model training, it should be noted that, of course, there are many more minor nuances and characteristics not discussed in this chapter. The chapter was primarily meant to give readers an idea of the environment at Dr. Pfleger and the underlying distributions of the used features and target values. Furthermore, the chapter provides an understanding of the distribution characteristics of the various features and target values, which is necessary for further model development and evaluation steps. An overview over all columns in the final feature set is shown in Table 8.

Table 8: Features Overview and Description

Technical Feature Name	**Description**	**Data Type**	**Feature Type**
ProductCode	The product code of the product to be produced in the associated order.	String	Categorical
Previous_ProductCode	The product code of the product produced in the previous order.	String	Categorical
Code	The production line as string on which the order is produced.	String	Categorical
CALC_WIRKSTOFF	Calculated feature describing the primary ingredient of the product.	String	Categorical
CALC_ALUFOLIE	Calculated feature describing the packaging foil used for the product.	String	Categorical
OrderQuantity	The amount of product units to be produced in this order.	Integer	Numeric
FS_Breite	Product Folding box (Faltschachtel) width in mm.	Integer	Numeric
FS_Länge	Product Folding box (Faltschachtel) length in mm.	Integer	Numeric
FS_Tiefe	Product Folding box (Faltschachtel) depth in mm.	Integer	Numeric
PBL_Breite	Product Package leaflet (Packungsbeilage) width in mm.	Integer	Numeric
PBL_Länge	Product Package leaflet (Packungsbeilage) length in mm.	Integer	Numeric
Tuben_Durchmesser	Product Tube diameter in mm.	Integer	Numeric
Tuben_Länge	Product Tube length / height in mm.	Integer	Numeric
CALC_PACKGROESSE	Calculated feature describing the amount of units per product package.	Integer	Numeric
10th_Percentile_Auftragswechsel	Calculated feature of the 10th percentile of the product changeover time (Auftragswechseldauer).	Float	Numeric
10th_Percentile_Primär	Calculated feature of the 10th percentile of the primary error time (Primärfehlerzeit).	Float	Numeric
10th_Percentile_Sekundär	Calculated feature of the 10th percentile of the secondary error time (Sekundärfehlerzeit).	Float	Numeric

5 Model Development

The training of machine learning models involves different algorithms, interfaces, and data preparation methods. This fact was discussed in the theoretical background. In the present project, the supervised learning technique family algorithms were used to solve the present regression problem of predicting several KPIs related to the OEE. Furthermore, as shown visually in the chapter 4.5 distribution charts and statistical characteristics of the target values, they are approximately normally distributed, which is a requirement for the MAE and (R)MSE to yield good results as shown in chapter 2.4.2. Therefore, the MAE and (R)MSE are suitable loss functions for the task at hand. Because the to-be-predicted KPIs are very sensitive to large errors and outliers, the RMSE was chosen as the primary loss function for the model training in the present project. The following chapters will describe the ML-methods and algorithms used to train several models for different target values. Briefly, those prediction target values will be described, the selection of used algorithms motivated, the chosen dataset splits, and the encoding and scaling methods discussed. Next, the results of the model training in different setups are given and analyzed. Finally, the last subchapter discusses some examples for the explainability of ML-models.

5.1 Model Targets

Target value columns are called training or model targets from here on out. In the present project, there are two types of model targets. Firstly, the KPIs OEE and its direct components, which are percentage values represented in a decimal range of $\{0, 1\}$. Secondly, the three planning related time KPIs, (unscheduled) downtime, the operating time, and the planned production time, which are represented in decimal hour range of $\{0, \infty\}$. These model targets have been selected based on the need to solve the business problem formally defined in the project requirements. The OEE and its KPIs are necessary in order to fulfill the primary objectives of the project and answer the research questions defined in chapter 1.2. At the same time, the time-related targets have been primarily used for additional requirements and the frontend user interface.

Table 9: Model Target Values Overview

Target	Long Name	Primary	Unit	Range
OEE	Overall Equipment Effectiveness	yes	percent	{0, 1}
PE	Performance Efficiency	yes	percent	{0, 1}
AR	Availability Rate	yes	percent	{0, 1}
QR	Quality Rate	yes	percent	{0, 1}
DT	(Unscheduled) Downtime	no	hours	$\{0, \infty\}$
OT	Operating Time	no	hours	$\{0, \infty\}$
PPT	Planned Production Time	no	hours	$\{0, \infty\}$

All model targets are predicted on the order level. In practice, this means predicting the OEE, its components, and the time KPIs for a not yet executed production order with the given input characteristics. The same loss function RMSE is used to train all model targets. Furthermore, all models are trained with the same input data and feature

sets. Finally, each target's models of different types are trained to determine the best performant model type for the present task.

As discussed extensively in previous chapters, the OEE is a composite or compound metric calculated by multiplying the availability rate, the performance efficiency, and the quality rate. Therefore, it was deemed interesting by the stakeholders, Dr. Pfleger and the author, to test if the predictions of models for the respective submetric target values could be multiplied to obtain a similar result as the directly learned OEE model, which can be considered an experiment and validation of the performance of the submetric models. This composite calculation is an artificial target value.

5.2 Selection of models for evaluation

For each model target several types of ML-algorithms, from here on out referred to as just models, are trained. As discussed in the second chapter, an extensive body of literature revolves around using machine learning techniques to improve production planning and control and more specific research on improving the OEE. The selection of models was inspired by the existing literature and similar case studies related to improving the OEE with the help of machine learning algorithms. Primarily, the meta-study of Usuga Cadavid et al. (2020), and the case study of El Mazgualdi et al. (2021) was used as inspiration for choosing a set of algorithms. Additionally, other related case studies and papers not necessarily doing a regression-based OEE prediction, but other approaches to using machine learning to improve OEE or production planning and control have inspired the set of selected model types (Senapati et al., 2024; Vilela De Souza et al., 2022; Dobra and Jósvai, 2022a,b; Lucantoni et al., 2023).

The final selection of methods is shown in Table 10 and has been sorted into the technique families defined by Usuga Cadavid et al. (2020) in their meta-analysis. They are either simple statistical learning or supervised machine learning models. The algorithms applied in this paper range from very simple regression techniques, like linear, polynomial, and ridge regression, to more complex algorithms like SVR, ensemble boosting learners, and simple multi-layered Neural Network (NN). The more straightforward regression methods are included in the set to provide a baseline against which the more complex techniques can be evaluated. Various techniques with different complexities and

Table 10: Applied ML-algorithms by their technique family after Usuga Cadavid et al. (2020)

Short Name	**Long Name**	**Technique Family**	**Technique**
LR	Linear Regressions	Regression	Simple Regression
RR	Ridge Regression (L2)	Regression	Simple Regression
PR	Polynomial Regression (Degree 2)	Regression	Simple Regression
DTR	Decision Tree Regression	Decision Trees	Decision Tree
RF	Random Forest Regressor	Ensemble Learning	Bagging
XGB	eXtreme Gradient Boosting	Ensemble Learning	Boosting
CatBoost	CatBoost - Gradient Boosting	Ensemble Learning	Boosting
LGBM	Light Gradient-Boosting Machine	Ensemble Learning	Boosting
SVR	Support Vector Regression	Support Vector Machines	Support Vector Machine
NN	3-Dense Layer a 20 Neurons Neural Network	Neural Networks	Neural Network

approaches have been chosen to provide a proper base for evaluating the best method to solve the present regression task. Chapter 2 provided a background understanding of all algorithm technique families, but not in detail for each algorithm. For detailed documentation the author refers to the algorithm packages used and stated in the Appendix. Most models have been used in their implementation by the Sci-Kit Learn Python package. All techniques have been trained using either the models' standard parameters or hyperparameter tuning, where applicable, as described in chapter 5.5.

The neural network is a straightforward 3-layered network consisting only of dense layers with 20 neurons each. No more complex model architectures have been attempted because this standard setup worked well or better than the other algorithms and seemed primarily limited by the available data. The neural network uses the Adam optimizer and RMSE as a criterion.

All of the selected methods require purely numeric inputs. Therefore different methods of encoding the categorical features had to be used. The SVR and the NN standard normalization was applied on the training data as briefly discussed in chapter 5.4.

5.3 Dataset Splits

This project uses a commonly applied technique in machine learning to split the available dataset into different subsets for training and validation. This split improves model generalization and allows for model validation on a model-unknown dataset to improve the validity of the findings and check for overfitting.

In the present project, a twofold approach was used to split the dataset. First, the data was split into a training and validation set with a given ratio. The training dataset is split into five folds using the K-fold cross-validation technique within the model training and optimization process. At the end of the model training pipeline, the performance evaluation of each trained model is done against the unseen validation set.

Different ratios for the validation set size have been selected to perform an experimental test. The chosen ratios are shown in Table 11. This experiment aims to evaluate which ratio of data for training versus validation results in the best model performance. The hypothesis behind these experiments is that a higher validation set ratio might result in lower model performance due to the limited number of available training examples. A ratio that is too low might, on the other hand, cause overfitting or similar issues.

Table 11: Validation Set Ratio - Experiments

Experiment Number	Ratio Validation Set
1	30%
2	25%
3	20%
4	15%
5	10%
6	5%

A randomized shuffle with the same functional implementation is used for each dataset split to ensure statistical validity across models and not introduce bias into the training

or evaluation due to the method used for the dataset split. Furthermore, to ensure that each model training result is comparable, a fixed random state for the dataset split was used in the training pipeline and for all experiments. This fixed random state results in randomization, always creating the same output of dataset split.

5.4 Data Encoding and Scaling

In order to make the machine learning algorithms trainable on the available dataset, feature columns in categorical form, in this case strings, need to be encoded. As discussed in chapter 2.4.9, multiple available encoding methods exist. The encoding methods have been selected based on the nature of each feature. Only the standard and straightforward encoding methods have been tested and applied. The selection of features to be encoded and the respectively used method are shown in Table 12.

Table 12: Applied Encoding Method per Categorical Feature

Feature	**Categorical Type**	**Encoding Method**
Product Code	Ordinal	Label / Ordinal Encoding
Previous ProductCode	Ordinal	Label / Ordinal Encoding
Active ingredient	Ordinal	Label / Ordinal Encoding
Packaging Foil	Ordinal	Label / Ordinal Encoding
Production Line	Nominal	One-Hot-Encoding

The (previous) product code is the unique identifier for each type of product stored as a string of a six-digit number. Product codes are ordered by the type of product, where similar products have similar identifying numbers. The more similar the product codes are, the more similar the products usually are. Though it is not a perfect ordinal representation, it was decided that the product codes are considered ordinal features and, therefore, encoded using ordinal encoding. Label encoding was also tested for the product code, but no relevant performance differences were found (compare chapter 6.2). The active ingredient and packaging foil similarly refer to identifying material numbers and exhibit the same characteristics as the product codes. The production line, on the other hand, is simply a string containing the short form name of the line, which makes it a nominal categorical feature and is represented with the one-hot-encoding technique.

As mentioned, the models for SVR and NN have problems training on datasets with vastly different column scalings, which can be remedied with feature scaling. Standard scaling is applied to all features in the training pipelines of these models, which zero-centers and standardizes the data (compare chapter 2.4.9). Scaling adjustments are only applied after the features have been encoded. Only the features have been standard scaled, not the target values. In another experiment, the standard scaling was applied to all model types, even those that do not require it, to test for a potential positive or negative effect on the model performance.

5.5 Hyperparameter Optimization

In improve the models' performance further, the method of hyperparameter optimization was used to enhance the model setup. Three types of hyperparameter tuning methods

have been employed throughout the one-year implementation timeframe: classic grid hyperparameter search, random hyperparameter search, and distribution-based hyperparameter search. These concepts have been briefly introduced in chapter 2.4.11, and due to similar optimization outcomes and significantly faster execution times, the final optimization run only used the distribution-based hyperparameter search. The hyperparameter search was done by implementing the Optuna package of Akiba et al. (2019), which they describe as an advanced and efficient framework to optimize hyperparameters.

Table 13: Hyperparameter Grids for Optuna Tuning

Algorithm	**Hyperparameter Grid**
RR	alpha: FloatDistribution(0.01, 10)
PR	degree: IntDistribution(2, 4)
DTR	max_depth: IntDistribution(10, 40)
	min_samples_split: IntDistribution(2, 11)
	min_samples_leaf: IntDistribution(1, 5)
RF	n_estimators: IntDistribution(50, 500)
	max_depth: IntDistribution(10, 50)
	min_samples_split: IntDistribution(2, 11)
	min_samples_leaf: IntDistribution(1, 5)
	bootstrap: CategoricalDistribution([True, False])
XGB	n_estimators: IntDistribution(50, 500)
	max_depth: IntDistribution(3, 10)
	learning_rate: FloatDistribution(0.01, 0.2)
	subsample: FloatDistribution(0.6, 1.0)
	colsample_bytree: FloatDistribution(0.6, 1.0)
	gamma: FloatDistribution(0, 0.5)
SVR	C: FloatDistribution(0.1, 10, logTrue)
	epsilon: FloatDistribution(0.01, 0.5, logTrue)
	kernel: CategoricalDistribution(['rbf', 'linear'])
LGBM	n_estimators: IntDistribution(50, 500)
	learning_rate: FloatDistribution(0.01, 0.2)
	num_leaves: IntDistribution(31, 255)
	boosting_type: CategoricalDistribution(['gbdt', 'dart', 'goss'])
	colsample_bytree: FloatDistribution(0.6, 1.0)
	reg_alpha: FloatDistribution(0, 1)
	reg_lambda: FloatDistribution(0, 1)
CatBoost	iterations: IntDistribution(100, 500)
	learning_rate: FloatDistribution(0.01, 0.2)
	depth: IntDistribution(4, 10)
	l2_leaf_reg: IntDistribution(1, 9)
	border_count: IntDistribution(32, 255)
	grow_policy: CategoricalDistribution(['SymmetricTree', 'Depthwise', 'Lossguide'])
NN	num_epochs: IntDistribution(100, 2000)
	learning_rate: FloatDistribution(0.001, 0.1)
	early_stop_patience: IntDistribution(25, 500)

Table 13 shows Optuna was used to check an extensive set of hyperparameter distributions for most employed algorithms. For linear regression, no specific hyperparameters can be tuned, and therefore, no optimization is to be done. Similarly, polynomial regression has no classical hyperparameters, but its capacity can be optimized for the underlying by attempting different polynomial bases.

For the defined neural network, only the number of epochs, the learning rate, and the early stopping patience have been optimized with hyperparameter grids. The architecture of the network was not changed in the optimization process; neither were the network optimizer, the activation functions, or potential dropout layers. It should be noted that multiple simple NN architectures have been tested in the implementation process with no noticeable performance differences between them. Therefore, it was decided to use the most straightforward setup.

The parameter grids have been chosen based on the proposed most important parameters in each algorithm's implementation documentation and on the authors' previous experience and therefore are not objectively defined. They are not meant to represent a complete optimization of all possible parameters but the most crucial ones for each algorithm.

In chapter 6 the results of the hyperparameter optimization will be shown for each model. These results will be shown only for one setup of training and validation run with the setup yielding the best performant result from the other previously discussed experiments.

5.6 Data, Training and Evaluation Pipeline

The previous chapters outlined the entire data pipeline, from raw data loading to data integration to model training processes. The development of this pipeline followed a modular and object-oriented programming approach, combining Python classes to implement the pipeline functions while a Jypyter notebook executes them. Figure 27 gives an overview of the finally created pipeline and the complete test setup used for the evaluation. Please note that this pipeline was executed for each experiment with the same input data but using the settings outlined in each experiment.

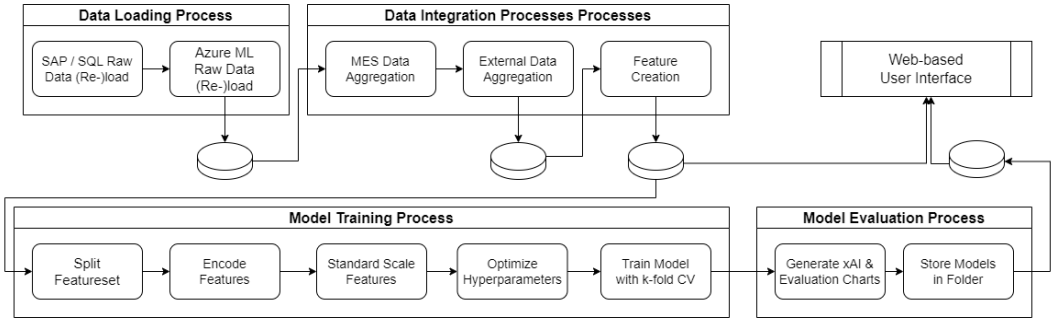


Figure 27: Final Project Data Pipeline Overview

It should be noted that the pipeline can either be triggered in its entirety, or each pipeline step can be executed individually by triggering each notebook separately. The first pipeline steps regarding data loading, integration, and feature creation are triggered once in this specified order for the test and validation setup. Each pipeline step that generates or transforms data stores its results as timestamped parquet files in a respective named

folder, shown as database slices in the figure. The subsequent processing step loads the file with the latest timestamp from the respective folders. Finally, the user interface draws from the generated data slices and the stored trained models to allow users interaction with the system (compare chapter 5.8).

For the evaluation setup, the model training process is executed several times with different setup parameters to obtain different results for each experimental setup and one final evaluation setup. Within each process execution, all models for the specified target value and model type are trained, evaluated, and stored. The models are stored as files inside a specified folder, inserting the name of the training run, the model's performance as RMSE, the model target, and the type inside the filename. This naming convention allows for easy sorting and loading of models of certain types or with the best performance for a specific target, a functionality used for the front end. Additionally, the pipeline provides validation and explainability charts for each model trained based on a provided verbose setting, which will be shown in chapter 6.

This pipeline setup is intuitive and easy for the company to use in productive operations. However, it is still crude due to its reliance on simple Jypyter notebooks triggering each other. It could be optimized using the Azure in-built pipeline feature and feature set storage. Furthermore, ideally, the models would be registered in the Azure ML environment with their respective characteristics to allow access to models from outside endpoints. These topics of deployment and maintenance have not been the focus of this project and, therefore, have not been implemented.

5.7 Model Ensembling and Explainability

The present project has trained numerous models using different machine learning algorithms. All models inherently make some prediction errors. In order to test if ensembling different model types can improve performance, an error correlation analysis was done. This test evaluates whether or not the models make the same errors or if they make different errors by calculating the correlation coefficient between each model's errors. Depending on the analysis's outcome, different models could potentially be ensembled if they tend to make different errors. In the other case that the models make the same errors overall ensembling is no viable strategy to enhance performance. If this were the case, it would indicate that the available data is the limiting factor for further performance improvements.

As discussed in chapter 2.4.12, explainability is critical for stakeholder acceptance of machine learning models. Therefore, the best-performing model was analyzed using the explainability method of SHAP after training the models. This exploration of the explainability method was not a focus of the research in this thesis but part of the practical project. It will, therefore, be briefly described as the example of the best performing OEE model. For this, the respective validation dataset of the model was used to the appropriate SHAP explainer class and generate the SHAP-values. These values were then displayed in several charts to answer the stakeholders' questions. Firstly, a bar chart that shows the average absolute impact of each feature on the model output explains the average influence of features on the model's predictions. Secondly, a scatter plot is used to show a feature's directional (instead of absolute) impact based on its value. Lastly, a more detailed analysis was done on the critical features to understand the direct learning

behavior in relation to that feature's values. The results of the explainability approach will be briefly outlined in the explainability section.

5.8 Web-based User Interface

For practical usage of the model inside the production planning process, it was decided to build a User Interface (UI) mockup for the planners. This interface was decided to be web-based, as most modern data tools are, and provide the ability to use the models to predict new, unplanned orders. This chapter will briefly explain the idea behind the UI, the implemented functionalities, and the technical tools used. It will not provide a detailed technical discussion about the UI's inner workings or web technologies used in creating it. It will especially explain how the user interface can be used to generate planning schedules even though the models can only predict the level of single orders. Please note that in figure 28 and 29, the user interface is in German because that is Dr. Pfleger's internal language.

For the mockup user interface, the Python package Flask was used. Flask is a Python package that implements a simple server-client architecture on a local computer or server compute instance. It allows to call Hypertext Transfer Protocol (HTTP) endpoints with get and post methods. In its standard functionality, Flask allows the use of a predefined Hypertext Markup Language (HTML) document that can be enriched with data from the backend element of Flask and return user inputs to the Flask backend with the HTTP post method (Flask, 2024).

In the present mockup, the Flask backend was used to load trained models and reference datasets and allow users to interact with the models by providing the required inputs using a combination of user inputs and reference data to predict the KPIs. Reference data here means features used for training, not input by the user because they are not commonly changed or unknown at the time of prediction. Examples for the first are the packaging dimensions, which are loaded from the reference dataset based on the product code provided by the user. Features unknown at the time of prediction are, for example, the order changeover time for which assumptions have to be made.

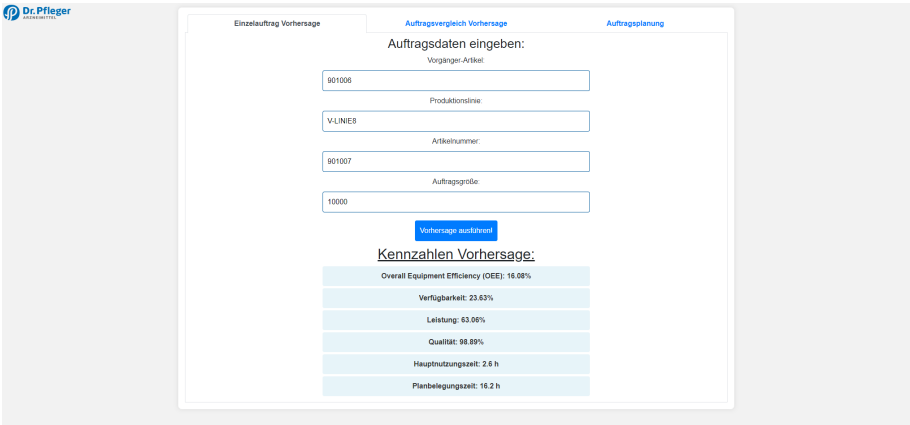


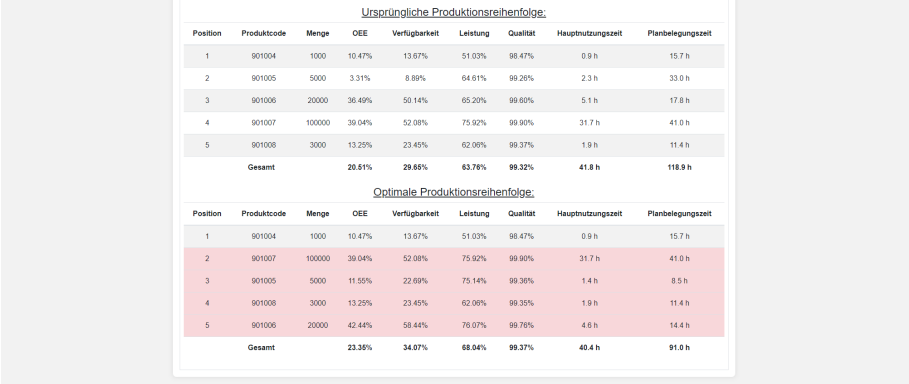
Figure 28: User Interface for single order prediction

The first iteration of the web interface allowed the user to execute a single order prediction by providing the inputs: previously produced product (id), production line, product to be

produced (id), and the order quantity. As described above, these provided user inputs are insufficient to predict with the trained models as they do not contain all features used for training. Therefore, the backend of the UI loaded all necessary features for a prediction: the product information, the packaging dimensions, the (historical) order changeover, and error time percentiles. The base assumption of the front end and the models is that the characteristics of future product changeovers are similar to the past. Not all potential combinations of product changeovers happened historically. For those with no historical reference data, the company predefined machine setup matrix has been used, which has been discussed in chapter 3.1 and the predefined times shown in Table 3. A limitation of this setup matrix is that it only includes products deemed necessary by the planners. For all product changeovers not present in the historical data or the setup matrix, the average times over all products were used as input for the models. These average times might differ greatly from reality and, therefore, triggers the UI to give the user an appropriate warning message. Finally, the display would show the user the predictions for all model targets defined in chapter 5.1 resulting from the best-performing model for the respective target.

After the first iteration of the mockup was completed and presented to the planners, it became clear that their priorities were diverging from the original requirement. New ideas and requirements were discussed about providing planners with scheduling proposals. As discussed in chapter 3.2.1, this represents the second step in Dr. Pfleger's vision towards a fully autonomous resource planning system. The company's stakeholders prioritized the topic over further optimization of the existing models. Therefore, an initial implementation was created within the mockup that allows the user to input a set of planned orders, where each order is given in the form of a product code and an order quantity. Additionally, the user must provide, as before, a previously produced product code and a line on which to produce. The set of to-be-scheduled orders is sent to the backend, and a finished planning proposal is returned and displayed by the system to the user.

Generating such a planning proposal is a challenging task. Initially, a simple greedy search was implemented, which was given the set of to-be-planned orders and iterated as long as any order remained to be planned. The provided order planning set resembles one production campaign discussed in chapter 3.1.1. At each iteration, each order in the set to be planned was predicted based on the previously produced product. The best order by the chosen KPI in each iteration is chosen as the next order, and the product of the current order is saved as a previously produced product. According to the model predictions, this greedy search was set to maximize the OEE to reach the most efficient production schedule. This search type was deemed insufficient for practical application because it tends to find only local optimal points in the solution space and, therefore, might not find the globally optimal solution. For this purpose, further methods of generation were developed. First is the permutational search, which is trivial to implement but very costly in computation. It generates all possible order permutations in the provided planning set, then generates the predictions for all permutations and chooses the permutation with the overall highest total metric. Predicting all permutations here becomes impractical performance-wise with a planning set larger than five elements. Since production campaigns or planning sets typically are about ten to fifteen orders large, predicting all permutations becomes unfeasible. To solve this problem, a more efficient search algorithm was required and found in the Simulated Annealing (SA) algorithm.



Ursprüngliche Produktionsreihenfolge:

Position	Produktcode	Menge	OEE	Verfügbarkeit	Leistung	Qualität	Hauptnutzungszeit	Planbelegungszeit
1	901004	1000	10.47%	13.67%	51.03%	98.47%	0.9 h	15.7 h
2	901005	5000	3.31%	8.89%	64.61%	99.26%	2.3 h	33.0 h
3	901006	20000	36.49%	50.14%	65.20%	99.60%	5.1 h	17.8 h
4	901007	100000	39.04%	52.08%	75.92%	99.90%	31.7 h	41.0 h
5	901008	3000	13.25%	23.45%	62.06%	99.37%	1.9 h	11.4 h
	Gesamt		**20.51%**	**29.65%**	**63.76%**	**99.32%**	**41.8 h**	**118.9 h**

Optimale Produktionsreihenfolge:

Position	Produktcode	Menge	OEE	Verfügbarkeit	Leistung	Qualität	Hauptnutzungszeit	Planbelegungszeit
1	901004	1000	10.47%	13.67%	51.03%	98.47%	0.9 h	15.7 h
2	901007	100000	39.04%	52.08%	75.92%	99.90%	31.7 h	41.0 h
3	901005	5000	11.55%	22.69%	75.14%	99.36%	1.4 h	8.5 h
4	901008	3000	13.25%	23.45%	62.06%	99.35%	1.9 h	11.4 h
5	901006	20000	42.44%	58.44%	76.07%	99.76%	4.6 h	14.4 h
	Gesamt		**23.35%**	**34.07%**	**68.04%**	**99.37%**	**40.4 h**	**91.0 h**

Figure 29: Example of a generated scheduling proposal

The chosen Simulated annealing (SA) is a probabilistic optimization algorithm which is derived from metallurgy annealing process, in which the metals are cooled slowly to reach a stable, low-energy state. The SA algorithm mimics this natural and physical behavior by exploring the solution space of an optimization problem, in this case, our proposal generation problem, allowing both uphill and downhill moves to escape local optima while iteratively and slowly reducing the probability of accepting a worse solution as the algorithms "temperature" decreases. This approach makes SA particularly useful for finding near-global optima in complex, multimodal solution space, where all possible permutations of a given planning set are present. One key advantage of SA is its ability to avoid getting trapped in local minima, making it highly effective for combinatorial optimization problems such as this present scheduling problem. Even with the SA, searching the entire solution space for the near-optimal solution takes about one minute for a campaign of 10-15 orders. Still, that is a bearable performance that the stakeholders accept for such a system. As with the other algorithms, greedy and permutational search, the SA returns the best-found production schedule to maximize a chosen metric to the user. Here, two metrics have been tested. First again, the OEE and, secondly, the PPT. The latter is chosen as, for the planners, the time planned for production is most crucial in their everyday work, while for the company, an optimal OEE is most relevant for financial success. The optimization process of these two metrics can lead to slightly different results. However, according to manual validation, more OEE efficient schedules also require less production time and vice versa.

Generally, all the development revolves around generating production schedules; an additional effort was made as a prototype and included in the thesis to give the reader an idea of how such models can be employed in praxis. It is a partially completed work or solution requiring further research, more profound design concepts, and further implementation efforts. Finally, the quality of the generated scheduling proposals is hard to judge and depends highly on the overall model performance and discussed assumptions about historical continuity. Impression of the results of scheduling proposal generation will be briefly provided in chapter 7. This explanation above about the possible algorithms to generate scheduling proposals is considered an anecdotal answer to the sixth research question from section 1.2. Further research on the topic would be necessary to answer this question correctly. How this topic could be researched and developed further will be discussed in the chapter 8.

6 Evaluation

The present chapter provides the result of the model development to answer the remaining research questions stated in chapter 1.2. All models are trained with the dataset described in chapter 4.5 without any additional exclusions or transformation of data besides the encoding and scaling methods discussed. The linear regression models are considered the baseline for comparing the different models. The primary performance metric used in training, validation, and evaluation is the RMSE. The MAE and R2 coefficients are provided for additional context. Furthermore, some experimental setups will be described in detail, as outlined in the previous section. Finally, the chapter will first provide the results of the models for the primary prediction target OEE and afterward the secondary target values described in section 5.1. All results of performance metrics shown are calculated on the validation dataset.

6.1 Experiment - Validation Set Ratios

As discussed in chapter 5.3, before further training of models for evaluation, an experiment was conducted to determine the optimal dataset split. The result of this experiment can be seen in Table 14, which contains the RMSE values for the OEE and its subcomponents in an aggregated form. The min, mean, and max values refer to an aggregation of all model types trained in the experiment with the validation setting for the respective model. In each column, the lowest value is underlined for ease of interpretation. As the

Table 14: Model Performance per Validation Ratio Experiment

Validation Set Ratio	**OEE**			**PR**			**AR**			**QR**		
	min	mean	max	min	mean	max	min	mean	max	min	mean	max
30%	0.1256	0.1384	0.1636	0.1072	0.1227	0.1516	0.1209	0.1359	0.1594	<u>0.0201</u>	0.1262	0.9876
25%	0.1265	0.1399	0.1646	0.1053	0.1199	0.1452	0.1226	0.1360	<u>0.1582</u>	0.0203	0.0321	<u>0.0798</u>
20%	0.1288	0.1418	0.1692	<u>0.0996</u>	0.1207	0.1745	0.1220	0.1372	0.1583	0.0210	<u>0.0308</u>	0.0800
15%	0.1292	0.1421	0.1690	0.1039	0.1220	0.1563	0.1224	0.1375	0.1632	0.0205	0.0318	0.0845
10%	<u>0.1062</u>	<u>0.1242</u>	<u>0.1575</u>	0.1017	<u>0.1194</u>	0.1580	<u>0.1109</u>	<u>0.1273</u>	0.1612	0.0281	0.0357	0.0813
5%	0.1100	0.1325	0.1625	0.1044	0.1209	<u>0.1413</u>	0.1152	0.1347	0.1620	0.0277	0.0418	0.0850

table shows, there is no intuitive result for which experiment results in the overall best performance because different model targets show different behaviors. By majority vote of the mean RMSE values, the validation ratio of 10% yields the best results. Only for the QR are there larger deviations. The minimum value would show the actual performance boundary determined by the ratio. Here the 10% validation ratio still yields the best performance for OEE and AR, while the PE and the QR yield the lowest minimum values with the 20% and 30% validation set ratio respectively. Since the prediction and optimization of the OEE is the primary focus, the 10% ratio was chosen for the final model training round evaluated in the next chapter. It should be noted that generally, the differences between the various ratios are not large, and differences in performance might also be explained by the validation set not losing certain outliers due to different dataset splitting. The same validation set ratio will be used for all model targets to ensure comparability between the different models. With the dataset split determined, the scaling and encoding methods will be evaluated in the following chapter.

6.2 Experiment - Scaling and Encoding Effects

Chapter 5.4 outlined why and how encoding and scaling transformations are to be applied to the training data. Now, these decisions are evaluated with the results in the tables below, which contain the model performance for each experiment as RMSE. Table 15 shows for the models trained to predict OEE that, as assumed in chapter 5.4, SVR and NN benefit from the application of standard scaling. All other model types trained

Table 15: OEE Model Performance: With vs. without Scaling

	LR	RR	PR	DTR	RF	XGB	SVR	CatB.	LGBM	NN
Scaling Off	0.1447	0.1446	0.1209	0.1569	0.1193	0.1125	0.1701	0.1068	0.1062	0.5337
Scaling On	0.5337	0.8185	0.5346	0.2171	0.2110	0.3103	0.1161	0.2126	0.1951	0.1179
Difference	**0.3891**	**0.6738**	**0.4137**	**0.0602**	**0.0917**	**0.1977**	**-0.0541**	**0.1058**	**0.0889**	**-0.4159**

for the OEE target observe worse performance when scaling is applied. The neural network is especially strongly negatively affected by training on unscaled features. At the same time, the simple regression models Linear Regression (LR), Ridge Regression Regression (RR), and Polynomial Regression (PR) are strongly negatively affected by training on scaled features. The effect on neural networks has to be taken with caution because it is exaggerated by the early stopping mechanism, which seems to stop the training of the NN due to it being unable to converge properly. For further evaluation, this

Table 16: Mean Performance Across Multiple Targets: With vs. without Scaling

	LR	RR	PR	DTR	RF	XGB	SVR	CatB.	LGBM	NN
Scaling Off	0.1109	0.1109	0.0988	0.1282	0.0956	0.0926	0.1447	0.0869	0.0874	0.5631
Scaling On	0.7429	0.5699	0.6034	0.2009	0.1579	0.2247	0.1092	0.1666	0.1438	0.0978
Difference	**0.6320**	**0.4590**	**0.5046**	**0.0728**	**0.0623**	**0.1321**	**-0.0354**	**0.0797**	**0.0563**	**-0.4653**

test was done as mean over the model targets OEE, PE, AR, and QR, which is possible because they have the same underlying range of values between 0 and 1. Table 16 shows this test's result, which exhibits the same characteristics as the result for the OEE target alone. Therefore, for further model training, scaling will only be applied to the model types SVR and NN training runs, while all other model types will not use feature scaling.

In addition to standard scaling, feature encoding was used to transform categorical

Table 17: OEE Model Performance: Ordinal vs. Label Encoding

	LR	RR	PR	DTR	RF	XGB	SVR	CatB.	LGBM	NN
Ordinal	0.1447	0.1446	0.1209	0.1597	0.1193	0.1125	0.1161	0.1068	0.1062	0.1117
Label	0.1447	0.1446	0.1209	0.1572	0.1195	0.1125	0.1161	0.1068	0.1062	0.1131
Difference	**0.0000**	**0.0000**	**0.0000**	**-0.0025**	**0.0002**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**-0.0016**

features into numeric ones. As discussed in chapter 5.4 for the ordinal features, both label and ordinal encoding methods have been tested. The results of the tests can be

Table 18: Mean Performance Across Multiple Targets: Ordinal vs. Label Encoding

	LR	RR	PR	DTR	RF	XGB	SVR	CatB.	LGBM	NN
Ordinal	0.1109	0.1109	0.0988	0.1267	0.0957	0.0926	0.1092	0.0869	0.0874	0.1012
Label	0.1109	0.1109	0.0988	0.1292	0.0958	0.0926	0.1092	0.0869	0.0874	0.0948
Difference	**0.0000**	**0.0000**	**0.0000**	**-0.0025**	**-0.0001**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0064**

seen in Tables 17 and 18. The results in the tables confirm that the encoding method has little impact on the model performance. No model shows performance difference above the third decimal digit. In most cases, there are tiny performance advantages favoring ordinal encoding, but the author deems them too insignificant to allow for an objective decision. As a standard method for ordinal features, ordinal encoding will be used for the model evaluation runs.

6.3 OEE Models

In the following chapter, the results of the primary task in the present project, the prediction of the OEE, will be provided. For this purpose, several model types have been trained, and for the final evaluation, hyperparameters were optimized using the Optuna method with the parameter distributions described in chapter 5.5. The optimized hyperparameter configuration can be found in the Appendix.

Table 19 shows the results for all performance metrics discussed in chapter 2.4.2 for the selected model types. The table highlights in bold that the CatBoost model, in its optimized form, is the best-performing model to predict the OEE based on all three metrics. Interestingly, for the CatBoost model, the performance slightly worsens regarding the MAE due to hyperparameter optimization and getting outperformed by XGBoost. Generally, the boosting models CatBoost, XGBoost, and LGBM perform very similarly.

Table 19: OEE Models Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt)	0.1447	0.1446	0.1209	0.1561	0.1197	0.1161	0.1125	**0.1068**	0.1062	0.1137
RMSE (Optimized)		0.1446	0.1209	0.1309	0.1088	0.1159	0.1068	**0.1067**	0.1069	0.1134
RMSE (Difference)		0.0000	0.0000	-0.0252	-0.0109	-0.0002	-0.0057	**-0.0001**	0.0007	-0.0003
MAE (Non-Opt)	0.1099	0.1098	0.0843	0.1036	0.0810	0.0837	0.0781	**0.0716**	0.0728	0.0817
MAE (Optimized)		0.1099	0.0843	0.0925	0.0751	0.0826	0.0725	**0.0736**	0.0746	0.0808
MAE (Difference)		0.0001	0.0000	-0.0111	-0.0060	-0.0011	-0.0056	**0.0020**	0.0018	-0.0009
R^2 (Non-Opt)	0.6680	0.6681	0.7681	0.6136	0.7728	0.7863	0.7992	**0.8191**	0.8211	0.7949
R^2 (Optimized)		0.6682	0.7681	0.7282	0.8122	0.7870	0.8190	**0.8195**	0.8188	0.7959
R^2 (Difference)		0.0001	0.0000	0.1146	0.0393	0.0007	0.0198	**0.0004**	-0.0023	0.0009

Overall, it can be concluded that the hyperparameter optimization has a moderate positive effect on the model performance for several models like DTR, Random Forest (RF), and eXtreme Gradient Boosting (XGB). Other models like the Light Gradient-Boosting Machine (LGBM) and CatBoost show slightly worse performance after hyperparameter optimization, which overfitting problems might cause. Generally, the performance of all

more complex models is within a similar range, indicating that the data is the limiting factor for improved performance rather than the model type or its hyperparameter optimization.

Table 20 averages the performance within the technique (families) defined in section 5.2. From the table, the simpler technique families, as expected, perform worse than more complex models. Boosting is the best-performing technique but is closely trailed by Bagging.

Table 20: OEE Models Performance Evaluation - Model Family Comparison

Metric	Simple Regression	DTR	Bagging	Boosting	SVM	NN
RMSE (Non-Opt)	0.1367	0.1561	0.1197	**0.1085**	0.1161	0.1137
RMSE (Optimized)	0.1328	0.1309	0.1088	**0.1070**	0.1159	0.1134
RMSE (Difference)	-0.0039	-0.0252	-0.0109	**-0.0015**	-0.0002	-0.0003
MAE (Non-Opt)	0.1013	0.1036	0.0810	**0.0742**	0.0837	0.0817
MAE (Optimized)	0.0971	0.0925	0.0751	**0.0736**	0.0826	0.0808
MAE (Difference)	-0.0042	-0.0111	-0.0060	**-0.0006**	-0.0011	-0.0009
R^2 (Non-Opt)	0.7014	0.6136	0.7728	**0.8131**	0.7863	0.7949
R^2 (Optimized)	0.7182	0.7282	0.8122	**0.8184**	0.7870	0.7959
R^2 (Difference)	0.0168	0.1146	0.0393	**0.0053**	0.0007	0.0009

Analyzing the models' results shows visually good learning behavior. An example of the CatBoost model and the NN is shown in Figure 30, where the left chart is a regression plot with the actual values on the x-axis and the predicted values on the y-axis. Besides a few outliers, most individual orders are predicted to be close to the actual values. The second chart depicts the common training versus validation loss chart for neural networks and visually indicates the functioning convergence of the neural network training.

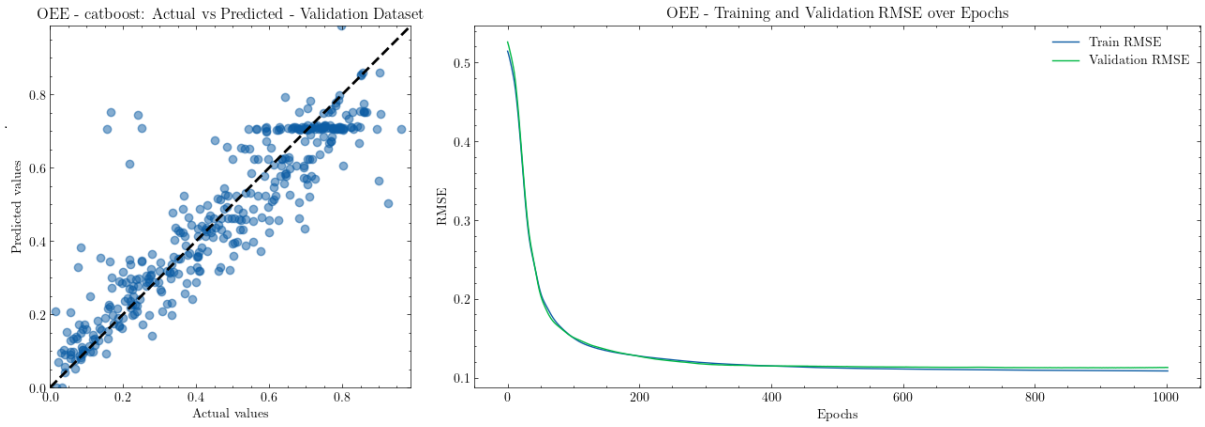


Figure 30: OEE Models Performance Evaluation - CatBoost and Neural Network Charts

In summary, it can be concluded from the above results that OEE can be predicted with the data in the present project but only with a moderate error margin in all tested model types. This error margin can result either from insufficient information density inside the data or from an unsuitable model training setup, which will be further evaluated in the

following sections about explainability and error correlation analysis. The fact that all model types perform pretty similarly, especially all more complex model types, indicates that the underlying data is the most likely root cause for the observed errors. The author considers this the answer to the third research question of chapter 1.2. The answer to the fourth question is that the Boosting models and, more specifically, the CatBoost model are the best performing on the dataset available at Dr. Pfleger.

6.4 OEE Submetrics Models

The following tables contain the results for models with the target values Availability Rate, Performance Efficiency and Quality Rate. Table 21 shows that the availability

Table 21: AR Models Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt)	0.1446	0.1446	0.1208	0.1622	0.1253	0.1152	0.1220	**0.1109**	0.1109	0.1151
RMSE (Optimized)		0.1446	0.1208	0.1246	0.1167	0.1156	0.1109	**0.1078**	0.1102	0.1125
RMSE (Difference)		0.0000	0.0000	-0.0375	-0.0086	0.0004	-0.0110	**-0.0031**	-0.0007	-0.0026
MAE (Non-Opt)	0.1091	0.1091	0.0869	0.1070	0.0863	0.0841	0.0855	**0.0768**	0.0780	0.0817
MAE (Optimized)		0.1091	0.0869	0.0887	0.0819	0.0815	0.0794	**0.0752**	0.0783	0.0768
MAE (Difference)		0.0000	0.0000	-0.0183	-0.0044	-0.0026	-0.0061	**-0.0016**	0.0003	-0.0050
R^2 (Non-Opt)	0.6438	0.7514	0.5520	0.7327	0.7740	0.7467	0.7905	**0.7905**	0.7744	0.7949
R^2 (Optimized)		0.7514	0.7354	0.7682	0.7726	0.7904	0.8021	**0.7932**	0.7844	0.7959
R^2 (Difference)		0.0000	0.1833	0.0355	-0.0014	0.0437	0.0115	**0.0027**	0.0100	0.0009

rate models perform very similar to the OEE models. The CatBoost model is the best-performing model, and the R^2 score shows a similar level of variance explanation. For the

Table 22: PE Models Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt)	0.1233	0.1232	0.1236	0.1507	0.1079	0.1245	**0.1078**	0.1017	0.1030	0.1293
RMSE (Optimized)		0.1232	0.1236	0.1190	0.1015	0.1236	**0.0987**	0.1003	0.1003	0.1204
RMSE (Difference)		0.0000	0.0000	-0.0316	-0.0064	-0.0008	**-0.0091**	-0.0014	-0.0028	-0.0089
MAE (Non-Opt)	0.0846	0.0846	0.0786	0.0948	0.0731	0.0900	**0.0735**	0.0693	0.0703	0.0904
MAE (Optimized)		0.0845	0.0786	0.0782	0.0690	0.0848	**0.0678**	0.0694	0.0686	0.0835
MAE (Difference)		-0.0001	0.0000	-0.0166	-0.0041	-0.0052	**-0.0057**	0.0001	-0.0017	-0.0069
R^2 (Non-Opt)	0.5034	0.5035	0.5005	0.2580	0.6194	0.4937	**0.6205**	0.6621	0.6530	0.4537
R^2 (Optimized)		0.5038	0.5005	0.5370	0.6632	0.5005	**0.6819**	0.6710	0.6714	0.5260
R^2 (Difference)		0.0002	0.0000	0.2790	0.0438	0.0068	**0.0614**	0.0089	0.0184	0.0723

PE, the best-performing model is XGBoost. Note, though, that the differences between the model types are smaller than for the previous two targets and that the R^2 score indicates that the models trained on the available data are worse at explaining the variance in the PE.

The models trained to predict the QR exhibit a few different characteristics than the models for the previous three targets. First, the general magnitude of error is smaller,

Table 23: QR Models Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt)	0.0312	0.0312	0.0300	0.0276	0.0283	0.0813	0.0283	**0.0281**	0.0296	0.0480
RMSE (Optimized)		0.0312	0.0300	0.0302	0.0283	0.0311	0.0292	**0.0286**	0.0294	0.0362
RMSE (Difference)		0.0000	0.0000	0.0026	0.0000	-0.0502	0.0010	**0.0006**	-0.0001	-0.0118
MAE (Non-Opt)	0.0118	0.0118	0.0105	0.0102	0.0093	0.0750	0.0094	**0.0088**	0.0091	0.0324
MAE (Optimized)		0.0118	0.0105	0.0094	0.0087	0.0119	0.0099	**0.0088**	0.0088	0.0200
MAE (Difference)		0.0000	0.0000	-0.0008	-0.0007	-0.0631	0.0005	**0.0000**	-0.0003	-0.0124
R^2 (Non-Opt)	0.1882	0.1882	0.2498	0.3646	0.3300	-4.5241	0.3327	**0.3419**	0.2691	-0.9260
R^2 (Optimized)		0.1882	0.2498	0.2368	0.3295	0.1909	0.2852	**0.3145**	0.2752	-0.0981
R^2 (Difference)		0.0000	0.0000	-0.1278	-0.0005	4.7150	-0.0475	**-0.0274**	0.0061	0.8279

and there are no significant differences between the simple baseline and more complex models. Additionally, the R^2 scores are generally relatively low, explaining only roughly 18 to 35 percent of the variance in the QR. Interestingly, the hyperparameter optimization seems to have no or a negative effect on all models except the SVR and NN. The QR models likely exhibit such lower error values because the overall variance in the metric is very low. One outlier model for the QR is the SVR. It performs poorly compared to the baseline linear regression model in its unoptimized form, as shown by the strongly negative R^2 score. SVR seems unable to fit the underlying distribution of the QR with its standard parameters, and only after optimization, reasonable performance is achieved. Similarly, the NN exhibits a negative R^2, indicating it performs worse than the baseline model linear regression. These results partially answer the fifth research question by showing that all the OEE subcomponents are predictable with moderate error margins. For a full answer to the fifth research question, chapter 6.6 evaluates the performance predicting the planning time KPIs. The varying performance on the different targets might arise due to the difference in underlying distributions of the metrics as discussed in chapter 4.5.

6.5 Experiment - OEE Composite Metric

Now that we have evaluated the models for OEE, AR, PE, and QR, we can experiment to test if the composite result of the models for AR, PE and QR predicts the OEE better than the trained model. For this, the predictions from each model and its respective type have simply been multiplied, and the result per model type has been compared to the predictions made by the best OEE model. Table 24 shows the error values for the composite results for the non-optimized and optimized model. The table also provides the difference between the optimized composite results and the trained OEE models. From those differences, it can be concluded that the composite metric performs slightly better for several model types than the trained OEE model. The NN is an outlier in that regard and performs significantly worse than the trained model. This decrease in performance for the NN might be caused by its bad performance for the QR. Finally, the CatBoost model is the best-performing one in the composite metric, and the composite metric is the best in general prediction.the OEE. One advantage of using the composite metric is that the predicted subcomponents and the OEE always align, which is not necessarily

Table 24: OEE Composite Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt)	0.1455	0.1455	0.1203	0.1498	0.1185	0.1285	0.1115	**0.1051**	0.1051	0.1202
RMSE (Optimized)		0.1454	0.1203	0.1177	0.1093	0.1152	0.1073	**0.1048**	0.1062	0.2014
RMSE (Difference)		-0.0001	0.0000	-0.0321	-0.0092	-0.0133	-0.0042	**-0.0003**	0.0011	0.0811
Composite Diff. to trained model	0.0000	0.0008	-0.0006	-0.0132	0.0004	-0.0007	0.0005	**-0.0019**	-0.0007	0.0879
MAE (Non-Opt)	0.1114	0.1113	0.0846	0.1006	0.0804	0.1000	0.0777	**0.0711**	0.0724	0.0859
MAE (Optimized)		0.1114	0.0846	0.0823	0.0752	0.0823	0.0741	**0.0715**	0.0730	0.1591
MAE (Difference)		0.0001	0.0000	-0.0184	-0.0051	-0.0177	-0.0037	**0.0004**	0.0006	0.0732
Composite Diff. to trained model	0.0000	0.0015	0.0003	-0.0102	0.0002	-0.0003	0.0016	**-0.0020**	-0.0016	0.0782
R^2 (Non-Opt)	0.6641	0.6644	0.7705	0.6441	0.7772	0.7381	0.8029	**0.8248**	0.8248	0.7707
R^2 (Optimized)		0.6646	0.7705	0.7802	0.8106	0.7894	0.8173	**0.8259**	0.8211	0.3569
R^2 (Difference)		0.0002	0.0000	0.1361	0.0334	0.0513	0.0144	**0.0011**	-0.0038	-0.4139
Composite Diff. to trained model	0.0000	-0.0036	0.0024	0.0519	-0.0015	0.0024	-0.0017	**0.0064**	0.0022	-0.4390

true for the trained OEE model. This experiment is considered successful and shows that the composite metric is as good if not better than the trained OEE model.

6.6 Planning Time Models

The following evaluation revolves around the prediction of the PPT, Operating Time (OT), and Down Time (DT). Those KPIs are measured in hours on the decimal scale. Due to the different value ranges, the errors for the three target values are not comparable with each other or between the models of OEE, AR, PE, and QR. These time values have been predicted to provide critical information for planning and a metric to generate planning proposals, as discussed in previous chapters.

Table 25: PPT Models Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt)	12.5334	12.5336	15.7067	17.7124	12.8374	12.2309	12.5632	11.7223	12.1641	11.7828
RMSE (Optimized)		12.5366	15.7067	12.8211	11.3516	11.3443	11.5137	11.3404	11.4932	12.0466
RMSE (Difference)		0.0030	0.0000	-4.8913	-1.4859	-0.8866	-1.0495	-0.3819	-0.6709	0.2638
MAE (Non-Opt)	6.8595	6.8581	5.8169	7.3061	5.6521	5.3804	5.6924	5.1960	5.4561	5.2737
MAE (Optimized)		6.8534	5.8169	6.1540	5.1426	4.7773	5.3416	5.2540	5.4084	5.7053
MAE (Difference)		-0.0047	0.0000	-1.1521	-0.5095	-0.6032	-0.3509	0.0580	-0.0477	0.4316
R^2 (Non-Opt)	0.4230	0.4229	0.0938	-0.1525	0.3946	0.4505	0.4202	0.4952	0.4565	0.4900
R^2 (Optimized)		0.4227	0.0938	0.3961	0.5266	0.5273	0.5130	0.5276	0.5148	0.4669
R^2 (Difference)		-0.0003	0.0000	0.5486	0.1320	0.0768	0.0928	0.0324	0.0583	-0.0231

Table 25 shows that the CatBoost model is the best-performing model to predict the PPT in regards to the RMSE and MAE. The R^2 scores explain roughly half of the variance in the actual values. Some models show considerable improvements in performance due to hyperparameter optimization, especially the DTR, while most show moderate improvements from the optimization. Only the model NN is an outlier in this regard

and shows worse performance from hyperparameter optimization. Generally, the simpler models perform similarly to more complex models, indicating that the available data is the limiting factor for better predictions. The more complex model performs very well

Table 26: OT Models Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt)	4.8642	4.8641	2.0556	4.6309	1.8944	3.3369	2.5558	2.6759	2.0390	1.7413
RMSE (Optimized)		4.8642	2.0556	5.2856	1.7439	2.3358	1.8824	2.1950	2.0966	2.1621
RMSE (Difference)		0.0001	0.0000	0.6547	-0.1505	-1.0011	-0.6734	-0.4809	0.0576	0.4208
MAE (Non-Opt)	3.7747	3.7763	1.2507	1.3496	1.0001	2.0653	1.1457	1.1239	1.1051	1.2329
MAE (Optimized)		3.7786	1.2507	1.4902	0.9495	1.5484	1.0807	1.0950	1.1253	1.6056
MAE (Difference)		0.0022	0.0000	0.1406	-0.0506	-0.5170	-0.0650	-0.0289	0.0202	0.3727
R^2 (Non-Opt)	0.6938	0.6938	0.9453	0.7224	0.9536	0.8559	0.9155	0.9073	0.9462	0.9608
R^2 (Optimized)		0.6938	0.9453	0.6384	0.9606	0.9294	0.9541	0.9376	0.9431	0.9395
R^2 (Difference)		0.0000	0.0000	-0.0840	0.007	0.0735	0.0387	0.0303	-0.0031	-0.0213

for the target OT as Table 26 shows. The models explain up to 95% of the underlying variance according to the R^2 scores. For OT, the RF model is the best performing, which is the first time in this evaluation that a nonboosting model is the best-performing model. As for PPT, several models show a mild performance improvement by the hyperparameter optimization process, while NN and DTR show a mild worsening. A performance improvement of factor two can be observed between simple regression and complex models in the RMSE and MAE. One explanation for the strong model performance when predicting OT might be that the operating time directly scales with the number of units produced.

Table 27: DT Models Performance Evaluation - Model Type Comparison

Metric	LR	RR	PR	DTR	RF	SVR	XGB	CatB.	LGBM	NN
RMSE (Non-Opt.)	10.7092	10.7098	15.1060	16.0693	12.3478	11.0713	12.6743	11.5708	11.1826	15.5991
RMSE (Optimized)		10.7188	15.1060	11.7096	10.9562	11.0455	10.8185	10.7337	10.9632	11.0355
RMSE (Difference)		0.0090	0.0000	-4.3596	-1.3915	-0.0258	-1.8558	-0.8370	-0.2194	-4.5636
MAE (Non-Opt.)	4.6019	4.6018	5.3069	5.9951	5.1828	3.9252	5.2535	4.7059	4.5890	9.5231
MAE (Optimized)		4.6065	5.3069	4.9635	4.5779	4.0141	4.6745	4.5838	4.7241	4.8436
MAE (Difference)		0.0047	0.0000	-1.0315	-0.6049	0.0889	-0.5790	-0.1220	0.1351	-4.6796
R^2 (Non-Opt.)	0.2487	0.2486	-0.4950	-0.6917	0.0011	0.1970	-0.0524	0.1229	0.1808	-0.5941
R^2 (Optimized)		0.2473	-0.4950	0.1017	0.2136	0.2007	0.2332	0.2452	0.2126	0.2022
R^2 (Difference)		-0.0013	0.0000	0.7934	0.2124	0.0037	0.2856	0.1223	0.0318	0.7963

Finally, Table 27 indicates the lousy performance of most model types predicting the DT. Especially without hyperparameter optimization, many model types perform worse than the linear regression baseline model, which, for some, even results in a negative R^2 score. None of the model types perform better in all metrics than the linear regression. Therefore, it is concluded that with the available data, the DT cannot be predicted with sufficient accuracy. Due to that fact, it is not usable in the user interface, and the models are not deployed.

The results above complete the answer to the fifth research question. All subcomponents of the OEE, the PPT, and the OT are predictable with the available dataset with mod-

erate error margins. Some targets might observe better performance than others due to either the data explaining them better or their underlying distributions differing.

6.7 Error Correlation Analysis

With multiple model types trained per target, one interesting evaluation is to check if the model's errors correlate. To this end, the correlation of errors between different models can be analyzed to determine the possibility of enhancing the performance through ensembling. However, if the errors of both models are strongly correlated, then ensembling might not be a viable strategy to improve performance.

One trivial example to understand the concept. If the RF predicts OEE to be 0.8 while XGBoost predicts 0.7 for a true OEE of 0.8, the ensemble prediction would average 0.75, which resembles an error of 0.05. Similarly, if both the models have highly correlated errors (for instance, both are giving an OEE of 0.7 for the actual value of 0.8), then ensembling them will not improve performance. For this purpose, a correlation matrix of the errors of all model types for each target has been calculated.

In this chapter, only the correlation matrix for OEE is analyzed in detail because it is the primary prediction target of the present thesis, and all other targets showed similar characteristics. As Figure 31 shows, all correlation coefficients range between zero and one. Furthermore, the figure shows that the errors of more complex models, especially

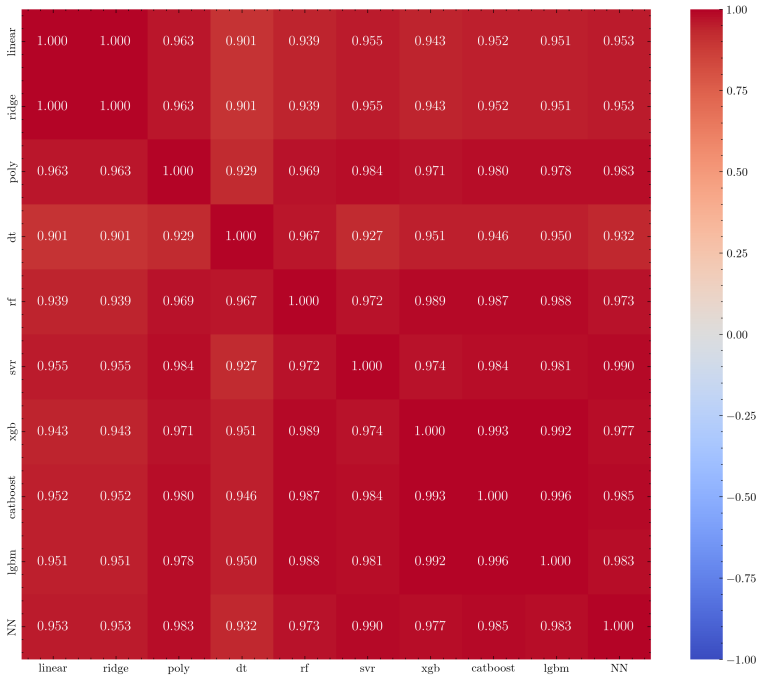


Figure 31: OEE Sample Error Correlation Matrix

the boosting models, strongly correlate with a coefficient of higher than 0.99. Some correlations are observed to be weaker, primarily the simpler models. This weaker correlation between simple and complex models is caused by the general performance difference between those models, where the simpler models observe worse performance. Even this weaker correlation has a higher coefficient than 0.9 in all cases.

From the correlation matrix, we can conclude that ensembling is not a viable strategy for the models trained in the present project to improve the performance of the OEE prediction.

6.8 OEE Model Explainability

To explain the model's decision-making to the stakeholders at Dr. Pfleger SHAP-values have been used. The concept behind SHAP values and explainability have been explained in the chapters 2.4.12 and 5.7. The following chapter provides an exemplary description of how xAI concept in the form of SHAP for the best performing OEE model, which is the CatBoost model. Throughout the project's timeline, explaining the OEE model as the primary target was the focus of explainability efforts. Other targets have only been spuriously explained with SHAP values. Three types of SHAP plots have been used for explaining: First, the mean absolute SHAP value per feature plot, which shows the average impact a feature has on a provided model's prediction over all analyzed instances. Secondly, a feature instance summary plot that allows the inclusion of the directional influence of a feature instance. Lastly, SHAP dependence plots were used to explain the learning behavior of a model in regard to a specific feature. Explainability plots used the model's validation dataset for the analysis.

Figure 32 contains the two mentioned summary plots against the entire used feature set. An explanation of each feature can be found in the Appendix. The left plot shows the features on the y-axis and the mean absolute SHAP values on the x-axis. According to the plot, the top three to five features dominate in their impact on the prediction. Especially, the order quantity and the product changeover time (10th_Percentile_Auftragswechsel) have strong average mean SHAP values, which is as expected. A surprise to the stakeholders was the strong impact of the packaging foil (CALC_ALUFOLIE_encoded) and the packaging width (FS_Breite) on the model's prediction. Furthermore, the plot shows

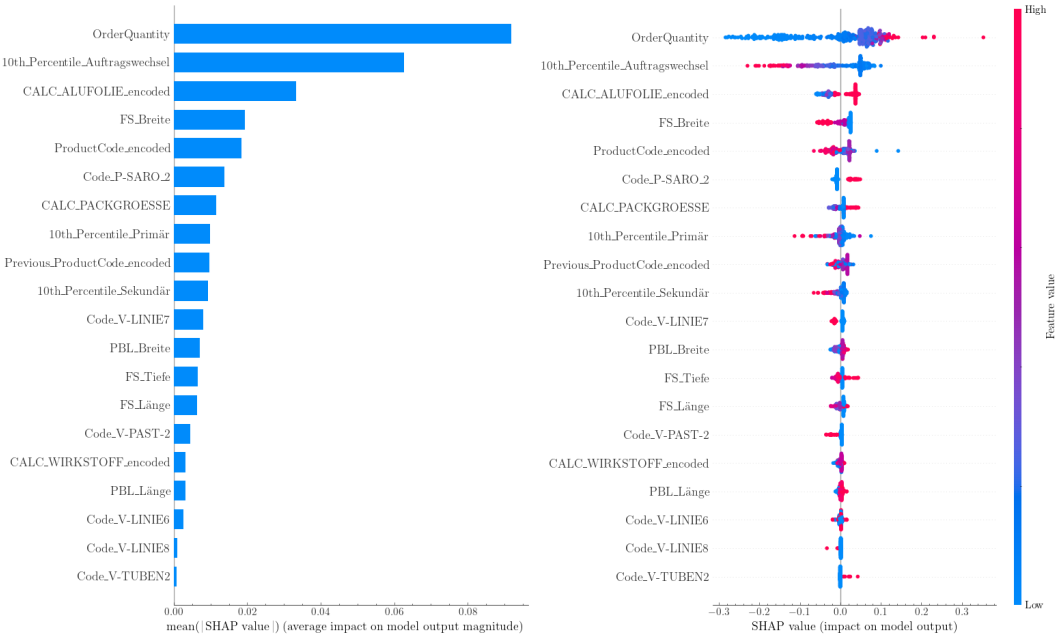


Figure 32: OEE Model Catboost - SHAP Summary Plots

that, as previously discussed, the different production lines impact the model's predic-

tions differently. Here the line with the strongest impact on the model's prediction is the P_SARO_2, while the blister and tube lines (V_Linie_6-8 and V_TUBE_2) show less impact on the prediction.

The right plot of Figure 32 shows the directional impact of a feature in relation to its value. Negative SHAP values must be interpreted as reducing the predicted value, while positive values increase the predicted value. As an example, it can be seen that the order quantity tends to improve the OEE when it is high, while it decreases it when it is low. This characteristic is reversed for the product changeover time, which decreases the predicted OEE when it is high but increases when it is low.

Not all present features exhibit such clear patterns; therefore, more in-depth explanations for features are necessary to understand the models' behavior. For this purpose, dependency plots for the most crucial features are used and shown in Figure 33. The dependency plots show one dot per feature instance, the impact as SHAP value on the y-axis, and the feature value on the x-axis. The first dependency plot shows the impact

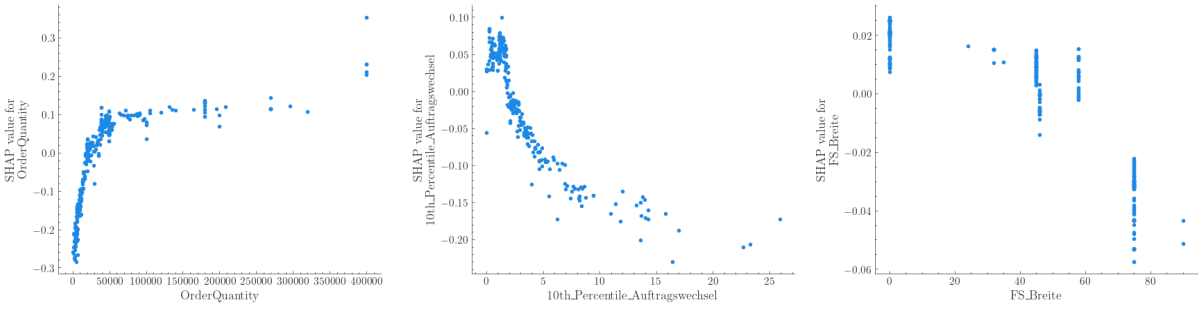


Figure 33: OEE Model Catboost - SHAP Feature Depency Plots

of order quantity on the prediction, approximating a logarithmic curve. This logarithmic characteristic means that in the range of the order quantity from 0 to about 50000, the impact on prediction changes very fast compared to the higher ranges between 50000 and 400000. An order quantity below about 25000 reduces the predicted OEE, while a value above that quantity increases the predicted OEE. Similarly, in the second plot of Figure 33, the product changeover time tends to increase the predicted OEE when shorter than roughly three hours. Longer product changeover times reduce the predicted OEE. Finally, the third plot shows less clear characteristics as the package width (FS_Breite) is no continuous variable, but only a few distinct widths. From this plot, we can conclude that it might not be learning correct behavior as for each distinct width, a highly varying impact on the predictions is observed. Generally, each distinct width has either a positive or negative impact on the prediction.

The analysis of the further features is not discussed in this chapter. Each feature has been briefly analyzed throughout the project's development time, and most showed expected learning behavior. The product packaging information and production lines are outliers and exhibit less clear learning behavior. These features will need to be analyzed further in the project's future development. The above discussion of applying the xAI methods highlights just one possible way to explain models to users and stakeholders. In this project, the explanations were crucial to create trust in the model's learning behavior.

6.9 Project Requirement Satisfaction

With all models and experiments evaluated, the final project requirement satisfaction can be analyzed based on the process outlined in Chapter 2 and 3. For this purpose, the company's stakeholders have been requested to assign a satisfaction and dissatisfaction score to each requirement. Satisfaction scores range between the values one and five. The result is shown in Table 28. In addition to the simple satisfaction and dissatisfaction score, the satisfaction considering the priorities has been calculated as the division of the score and the priority. The two metrics are not directly comparable, but the ratio between satisfaction and dissatisfaction can be compared. Furthermore, Table 28 shows in the col-

Table 28: Project Requirements Satisfaction and Dissatisfaction

N.	Primary	Priority	Completed?	Satisfaction	Dissatisfaction	Satisf.action cons. Priority	Dissatisfaction cons. Priority
1	Yes	1	Yes	5.00	1.00	5.00	1,00
2	Yes	1	Yes	5.00	1.00	5.00	1,00
3	Yes	1	Yes	5.00	1.00	5.00	1,00
4	Yes	2	Yes	5.00	1.00	2.50	0,50
5	Yes	1	Yes	5.00	1.00	5.00	1,00
6	No	2	Yes	4.00	2.00	2.00	1,00
7	Yes	1	Yes	5.00	1.00	5.00	1,00
8	No	3	Yes	5.00	1.00	1.67	0,33
9	No	3	Yes	5.00	1.00	1.67	0,33
10	No	2	Yes	2.00	4.00	1.00	2,00
11	Yes	1	Yes	5.00	1.00	5.00	1,00
12	Yes	3	Yes	5.00	1.00	1.67	0,33
13	Yes	1	Yes	5.00	1.00	5.00	1,00
14	No	4	Yes	4.00	2.00	1.00	0,50
15	No	5	Yes	4.00	2.00	0.80	0,40
16	No	2	No	2.00	3.00	1.00	1,50
17	No	5	No	1.00	3.00	0.20	0,60
18	No	5	No	1.00	2.00	0.20	0,40
19	No	5	No	1.00	2.00	0.20	0,40

umn 'Completed?' which of the requirements has been resolved. Some requirements were outside the original scope of this thesis and can be considered as future work. Of all 19 requirements, 15 have been completed. The mean satisfaction score is 3.89, and the mean dissatisfaction score is 1.63, which makes a satisfaction-to-dissatisfaction ratio of 2.39. From these scores, the overall high stakeholder satisfaction with the requirement completion can be concluded. Considering the priorities, this ratio rises to 3.2, indicating that

Table 29: Satisfaction and Dissatisfaction by Completion Status

Completed?	Mean Satisfaction	Mean Dissatisfaction	Mean Satisfaction cons. Priority	Mean Dissatisfaction cons. Priority
No	1.25	2.50	0.40	0.73
Yes	4.60	1.40	3.15	0.83

especially the highly prioritized requirements have been fulfilled with higher stakeholder satisfaction. When aggregating by requirement completion, it becomes apparent that the stakeholder dissatisfaction primarily originates from the not completed requirements, while the completed requirements enjoy a very high satisfaction score of 4.6 to dissatisfaction of 1.4 (compare Table 29). Similarly, Table 30 shows that primary requirements show higher satisfaction scores due to receiving more focus and attention throughout the development process. Functional requirements are also fulfilled with higher satisfaction

Table 30: Satisfaction and Dissatisfaction by Primary Flag

Primary	Mean Satisfaction	Mean Dissatisfaction	Mean Satisfaction cons. Priority	Mean Dissatisfaction cons. Priority
Yes	5.00	1.00	4.35	0.87
No	2.90	2.20	0.97	0.75

than non-functional requirements (compare Table 31). This project focused on building a functional prototype; therefore, functional requirements received more focus and development time. Lastly, Table 32 highlights that the requirements related to the pipeline

Table 31: Satisfaction and Dissatisfaction by Requirement Type

Requirement Type	Mean Satisfaction	Mean Dissatisfaction	Mean Dissatisfaction cons. Priority	Mean Satisfaction cons. Priority
Functional	4.75	1.25	0.70	3.03
Non-functional	2.43	2.29	0.99	1.80

steps 'Data Integration' and 'Model Training' have higher satisfaction scores than the 'Frontend' or 'Model Evaluation'. The user interface was a requirement added late in the project and received only limited attention. The Model Evaluation might exhibit lower satisfaction scores due to one primary evaluation requirement that could not be fulfilled: The real-world application of the system (Req. 16) to evaluate its effectiveness in the running planning process.

Table 32: Satisfaction and Dissatisfaction by Area of Pipeline

Area of Pipeline	Mean Satisfaction	Mean Dissatisfaction	Mean Satisfaction cons. Priority	Mean Dissatisfaction cons. Priority
Data Integration	4.17	1.33	3.28	0.82
Model Training	4.25	1.75	2.33	0.92
Model Evaluation	3.60	1.60	2.57	0.85
Frontend	3.00	2.33	0.67	0.50

In summary, the project was a success based on the evaluation in the previous chapters, the fulfilled requirements, and the overall high satisfaction scores. The stakeholder satisfaction is reasonable to very good, and all primary requirements were completed. The unfulfilled requirements or those with lower satisfaction scores highlight areas for further work and research possibilities.

7 Discussion

The present thesis described in the previous chapters a practical implementation project trying to optimize the OEE with the help of machine learning methods. Several ML algorithms have been used to train models to predict KPIs related to the OEE. The thesis discussed a nearly complete creation of a machine learning project from the raw data collection to the explainability of trained models. Therefore, it can inspire how such an end-to-end project can be conducted in practical environments. Furthermore, it highlights the challenges of working in practical settings with data availability and quality limitations. Finally, it shows the importance of testing several machine learning methods against a baseline method to analyze the performance results thoroughly.

The performance results of the model training are considered satisfactory for a prototype project but leave room for improvement for a future productive tool. The conducted experiments evaluated in the chapters 6.1 and 6.2 highlight the importance of evaluating certain tooling choices in applying machine learning methods. These experiments especially showed that for smaller datasets, a proper validation set ratio needs to be chosen and that different ML algorithms react quite differently to the usage of standard scaling. Furthermore, the results provided regarding hyperparameter optimization in chapter 6 highlight that different methods react differently to optimization and that optimization of one performance metric might worsen the model's performance in another (here RMSE vs. MAE). Overall, hyperparameter optimization seems to have a limited effect on the performance of trained models, and the standard parameters for most models are sufficient. One noticeable exception from this trend is the RF and DTR, which seem to benefit from hyperparameter optimization more than the other models for most targets. This more substantial improvement for the tree-based models might be explainable due to the standard model parameters allowing trees of unlimited size, which might lead to overfitting issues.

Generally, the projects' goals have been achieved, and the cooperation partner, Dr. Pfleger, is satisfied with fulfilling their requirements. The OEE could be predicted with the, aimed for, error margin of about 10%. The thesis highlighted the strength of ensembling methods in this use case, in which both Bagging and Boosting techniques outperformed most other simple techniques. This strong performance of ensembling methods is likely because they can handle outliers better than the other methods due to the nature of ensembling various differently trained models. A more extensive analysis would need to be done to understand why these model types perform better conclusively.

Additionally, this thesis highlights that even elementary regression methods perform relatively well on the given dataset and that more complex models yield only minor performance improvements for most model targets. While the models predicting the OEE and AR explain the underlying variance relatively well, models for the PE, the DT, and especially the QR are unable to fit the underlying variance. Note that many of the features used in training those models are known to affect primarily the OEE and AR, while it is unclear to the company's stakeholders what causes the variance in the QR. Furthermore, the error correlation analysis highlights that ensembling the different model types is unsuitable for improving performance. Those facts might hint that the only remaining path to improve model prediction performance further is using more, better, or additional data for training. The models might benefit from including more data besides the product packaging information or the pure MES data. For example, the used feature set does

not include information about the availability of materials, the involved steps of machine adjustments, workers in a shift, the expected delivery times, and similar topics outside of the management data scope. Critically, some data that would probably be crucial to predicting the OEE, like data regarding the staff involved in the production project, will, even if available, not be usable in such a project due to data privacy. Therefore, performance is, in practice, also limited by outside regulations.

Note the following additional limitations of the present project. First, data quality problems occurring while aggregating the MES data make a conclusive evaluation in the live process difficult and reduce the available training examples, which might negatively affect the performance. Furthermore, as shown and discussed in Chapter 6, the available data seems insufficient to predict the target KPIs with a lower error margin. Most models still show considerable errors even after hyperparameter optimization. This fact might prevent the practical application of the models trained in this project even though the defined performance goal of the stakeholders has been satisfied. The most crucial limitation of the present thesis is the lack of real-world application. The thesis did show that the models can predict past orders with moderate error margins, but does that mean they can predict future orders? This question, which is a short form of the seventh research question, could not be answered throughout this thesis due to the lack of trust of stakeholders and planners in using the system, the limitation of the implementation timeframe, and the limited time planners have to be closely involved in the project. Using the system in practice, in the everyday work of the planners, and evaluating the models' predictions against real-world observation would be a crucial next step in understanding whether or not the OEE can indeed be improved by the developed tool. Due to this How this could be done will be discussed in the next chapter. Closer cooperation with the end users and a greater effort to increase their trust with explainability methods might have been crucial to faster use of the system in practice. The methods used described in chapter 5.7 and 6.8 might not be extensive enough or follow a scientifically valid approach to provide model explanations with xAI methods. Further research would be required into other available methods like Local Interpretable Model-Agnostic Explanations (LIME) or model-inbuilt explainability methods like decision tree-based explanations.

The user interface built in this mockup highlights the need for a practical project to consider the deployment and operations of the models trained. An easily accessible and intuitive interface to trained machine learning models seems necessary to conclude such a project successfully. In the case of the present thesis, deployment, operations, and user interface have not been a focus. Nonetheless, they are critical for the project's long-term success at Dr. Pfleger. Operations in this context involve the regular retraining of models on new incoming data and constant performance monitoring. Deployment highlights how models are made available to users, which is usually done by a graphical user interface. An intuitive user interface might motivate stakeholders and users to interact, drive the project forward, and reduce resistance to introducing a ML-based tool into their everyday work life.

Lastly, the generation of scheduling proposals was an addition to the project's original scope that needs further research and work. Using models that can predict individual orders to construct scheduling proposals iteratively may not be scientifically valid or purposeful. Other machine learning methods that fit the nature of the proposal generation problem should be investigated. Possibilities might be a RL approach or neural network architectures that can predict sequences like Recurrent Neural Networks. The research

from Usuga Cadavid et al. (2020) seems to indicate that such RL are used frequently for similar problems. The order-based approach seems to yield satisfactory results from the initial evaluation but needs real-world evaluation to validate this subjective evaluation. Furthermore, it should be noted that this generation problem, whichever method is used, suffers from a lack of knowledge at the time of prediction about topics like delivery expectancy, material availability, supply chain stability, and staff availability. These factors would be crucial to creating a fully autonomous system for production planning and would require significant work to gather, validate, and prepare new data sources.

As the above discussion indicates, the project at Dr. Pfleger has many avenues and ways to progress further and invest more time to achieve the vision outlined by Dr. Pfleger. The next chapter will briefly outline how the research and project could be developed further.

8 Future Work

The main ways of further developing the project of the present thesis are data enrichment, real-world testing, and further research into generating scheduling proposals. This chapter will briefly describe further work possible in these three areas.

Data enrichment is a topic already mentioned in Dr. Pflegers long-term vision outlined in Figure 14 and revolves around opening up and using new data sources. This additional data should be chosen such that it has explanatory content about so far unexplained variance in the distributions of the KPIs related to the OEE. Most likely this would require the connection of new databases, tapping into new data in the existing SAP or MES data or including additional external manually created information. Crucial would be collecting and including data about the process in the cleanroom area of the production lines. Another interesting topic would be to find data that better explains the variance in the quality rate, meaning data that explains why specific production orders result in more scrapped products than others. Finally, new data explaining the occurance of unscheduled downtime better would be crucial to close the gap in predicting the downtime but also the AR and the OEE.

Real-world testing is critical to evaluate whether one of the core assumptions of the build tool holds: That the future behaves similarly to the past. To execute such a test, planners need to adjust their plans with the help of suggestions they get for the model. These adjustments could be limited in scope to one production line for ease of evaluation; for this line, production campaign planners could generate scheduling proposals over a defined timeframe. These schedules would then be executed throughout the timeframe, and their true KPIs recorded by the MES. These records could then be evaluated against the original predictions made by the models, which allows us to determine the differences between predicted KPIs and true KPIs. The resulting range of differences will determine whether the tool can be employed in the practical process.

Lastly, further research and development into how to generate good scheduling proposals should be conducted. To further develop the topic around scheduling proposal generation, extensive literature research should be conducted to check for similar problems that have already been solved. Several techniques might be found in this literature review, which can then be tested and evaluated against the existing method. Furthermore, other types of models should be trained and evaluated that have inbuilt abilities to predict sequences like recurrent neural networks or potentially a RL agent could be trained to solve the scheduling problems. These approaches differ significantly from the SL approach employed in this thesis and, therefore, would require more research in their theoretical background.

Finally, the practical application of the software package developed throughout the implementation phase of this project should be made possible with ML deployment and operation techniques. Models trained should be appropriately registered into the Azure ML platform, including their respective target, performance, and reference to the dataset version used for retraining. Furthermore, the user interface would need to be deployed on a productive server with proper security measures and connected to the company's single-sign-on system to provide easy access to users. A regular automatic retraining and validation pipeline could be implemented. Newly retrained models would be deployed to the user interface based on predefined logic regarding their monitored performance.

9 Conclusion

This thesis described a holistic approach based on the DSR methodology for improving the OEE through the application of machine learning techniques. Through the use of several ML algorithms to predict several KPIs and allow model interaction in a user interface, the thesis was able to show how ML can be used to enhance the efficiency of industrial processes, especially for production planning and control at Dr. Pfleger.

The main goal was to train a model that would allow to predict the OEE and its subcomponents with a certain level of accuracy. The thesis demonstrated that the models' performance is reasonable for OEE and AR prediction by employing various machine learning models, including ensemble models like bagging and boosting. These results show that ensemble methods are superior to simple regression models in this project, most likely, because they are better able to deal with data variability and outliers.

However, the models trained were less accurate in predicting other subcomponents of OEE, namely Performance Efficiency and especially Quality Rate. The reduced performance in predicting these subcomponents can most probably be attributed to the lack of sufficient data, which emphasizes a key challenge in this research: the need for further data sources to explain the variance in all KPI targets better.

As for the practical outcome, the project has given Dr. Pfleger the initial prototype that can predict the OEE with an RMSE of approximately 10%. This satisfies the defined performance objectives and provides a good basis for future work and possible real-world application. However, the models have to be tested for their effectiveness in real life situations to ascertain their accuracy in prediction. In addition, the trust of the stakeholders and the usability of the tool should be increased by refining the user interface and incorporating the system into the production planners' work processes.

Therefore, this thesis has provided evidence and possibilities of enhancing OEE in production systems using machine learning. Despite the fact that the project achieved its goals, there are several directions for further research: data enrichment, real-world experiments, and enhancing the generation scheduling proposals. These areas will be important in order to achieve the potential of machine learning in the improvement of production processes at Dr. Pfleger.

A Appendix

A.1 Code Repository

The code written to solve the task in the present thesis and to produce the results is made available under MIT license and can be used with the license terms at:

`https://github.com/Veheled/ML_Prediction_OEE`

The code repository contains an introductory readme file that explains roughly the functioning of each notebook and module as well as how to run them. Furthermore, the readme quickly explains how the user interface can be started and accessed. Note that this repository does not include any data due to the data being owned by Dr. Pfleger. The code expects a folder structure to function. Also it expects datafiles with a specific naming convention to be downloaded from an Azure Storage in the dataloader module. It was developed in the Azure ML Studio environment and expects certain native function and packages related to this environment to function. Furthermore, a set of external python packages are required for the code to run. They are specified in the requirements.txt and called for in the setup.py. The python package defined by the code in the repository can be installed via pip using the link above.

A.2 Requirement Descriptions

N.	Short Name	Type	Primary	Priority	Area of Pipeline	Description
1	Data Exploration and Business Knowledge Collection	Non-functional	Yes	1		Collect business insights and explore existing data sources to define project goals and requirements and understand how data needs to be integrated and connected for further analysis.
2	Implementation of Database Connection	Functional	Yes	1	Data Integration	Establish connections to necessary data sources from Azure data storage to load the MES raw data. The data should be loaded and stored using a timestamp versioning system to allow backtracking of which pipeline run produced and used which input data.
3	Data Integration and Aggregation of MES Raw Data	Functional	Yes	1	Data Integration	Integrate raw data from MES with the help of the provided datamodel to obtain the KPIs similarly as they are provided in the MES frontend tool. The result should return all KPIs relevant for the OEE calculation as well as all other necessary data contained in the MES raw data on the basis of an order.
4	Processing of Static Product and Packaging Information	Functional	Yes	2	Data Integration	Load, clean, process, and join the data provided via Excel files into the aggregated MES data via their associated product key. The information should be transformed from strings into numbers for fields containing multiple dimensional packaging information.
5	Data Validation and Quality Checks	Functional	Yes	1	Data Integration	Execute data quality tests and implement validation checks to ensure the quality and consistency of the integrated data with the frontend of the MES.
6	Feature Engineering for Model Training	Functional		2	Data Integration	Explore possibilities to engineer new features based on the MES data to improve model accuracy and performance.
7	Training of ML Models to Predict the OEE	Functional	Yes	1	Model Training	Develop a pipeline to train machine learning models that are capable of predicting the OEE based on the integrated dataset. This pipeline should follow the commonly used structure of ML training like a train-test split, validation on unseen data, and test various different types of models.
8	Training of ML Models to Predict Additional KPIs	Functional		3	Model Training	Train additional models to allow prediction of other KPIs among which the Availability, Performance, Quality, and planned production times.
9	Hyperparameter Tuning and Model Optimization	Functional		3	Model Training	Optimize models through hyperparameter tuning to improve their predictive accuracy.
10	Model Versioning and Experiment Tracking	Non-functional		2	Model Training	Implement a way to version control trained models, store their respective loss, and experiment tracking to manage model iterations and experiments.

N.	Short Name	Type	Primary	Priority	Area of Pipeline	Description
11	Model Evaluation and Selection of Best Approach	Functional	Yes	1	Model Evaluation	Evaluate the performance of all models trained and determine the best-performing model for a productive deployment.
12	Evaluate Method for Explainability of Model Decisions	Functional	Yes	3	Model Evaluation	Explore and implement methods to explain the model's predictions to decision-makers in business, highlighting key influencing factors on models' predictions.
13	Average OEE Prediction Error of Below 10 Percentage Points	Non-functional	Yes	1	Model Evaluation	If possible, the goal is to train models that can achieve to predict the OEE with an average error of 10 percentage points. The error range was chosen in regards to the assumed precision of prediction allowed by the data.
14	Implementation of User Interface Prototype for Planner Interaction	Functional		4	Frontend	Create a user interface for planners to interact with the trained models, inputting necessary parameters for prediction, drawing historic reference data and features, and displaying predictions returned by the model.
15	Implementation of Methods to Allow Planners to Get Optimized Production Sequences	Functional		5	Frontend	Create a user interface and associated background algorithm that allows planners to optimize input production sequences by using models to predict multiple combinations of possible sequences.
16	Real-World Performance Evaluation in Production Planning	Non-functional		2	Model Evaluation	Test the model in a real-world production environment to assess its effectiveness in optimizing order sequencing.
17	Integration with Existing Production Planning Tools	Non-functional		5	Frontend	Ensure seamless integration of the ML model with existing tools used by production planners.
18	Scalability and Performance Optimization of Pipeline	Non-functional		5	Data Integration	Optimize the data pipeline to handle large datasets and ensure efficient processing.
19	Continuous Monitoring and Feedback Loop	Non-functional		5	Model Evaluation	Set up monitoring tools to track model performance over time and incorporate feedback for continuous improvement.

A.3 Technical Feature Descriptions

Technical Feature Name	Description
ProductCode_encoded	The ordinal / label encoded product code of the product to be produced in the associated order.
Previous_ProductCode_encoded	The ordinal / label encoded product code of the product produced in the previous order.
OrderQuantity	The amount of product units to be produced in this order.
FS_Breite	Folding box (Faltschachtel) width in mm.
FS_Länge	Folding box (Faltschachtel) length in mm.
FS_Tiefe	Folding box (Faltschachtel) depth in mm.
PBL_Breite	Package leaflet (Packungsbeilage) width in mm.
PBL_Länge	Package leaflet (Packungsbeilage) length in mm.
Tuben_Durchmesser	Tube diameter in mm.
Tuben_Länge	Tube length / height in mm.
CALC_PACKGROESSE	Calculated feature describing the amount of units per product package.
CALC_WIRKSTOFF	Calculated feature describing the primary ingredient of the product.
CALC_ALUFOLIE	Calculated feature describing the packaging foil used for the product.
10th_Percentile_Auftragswechsel	Calculated feature of the 10th percentile of the product changeover time (Auftragswechseldauer).
10th_Percentile_Primär	Calculated feature of the 10th percentile of the primary error time (Primärfehlerzeit).
10th_Percentile_Sekundär	Calculated feature of the 10th percentile of the secondary error time (Sekundärfehlerzeit).
Code_P-SARO-2	One-Hot-Encoded feature whether or not the order is produced on the specific production line, which is the saro line.
Code_V-PAST-2	One-Hot-Encoded feature whether or not the order is produced on the specific production line, which is the pastille line.
Code_V-LINIE6	One-Hot-Encoded feature whether or not the order is produced on the specific production line. Line 6 is a blister line.
Code_V-LINIE7	One-Hot-Encoded feature whether or not the order is produced on the specific production line. Line 7 is a blister line.
Code_V-LINIE8	One-Hot-Encoded feature whether or not the order is produced on the specific production line. Line 8 is a blister line.
Code_V-TUBEN2	One-Hot-Encoded feature whether or not the order is produced on the specific production line, which is the tube line.

A.4 KPI Descriptions

KPI	Tables Involved	Calculation Description
Good Quantity	FactMdaCounter, DimMdaCounter, DimWorkcenter	Sum of values where Kind is "OK", considering if Quality is enabled.
Rework Quantity	FactMdaCounter, DimMdaCounter, DimWorkcenter	Sum of values where Kind is "Rework", considering if Quality is enabled.
Scrap Quantity	FactMdaCounter, DimMdaCounter, DimWorkcenter	Sum of values where Kind is "NOK", considering if Quality is enabled.

KPI	Tables Involved	Calculation Description
Target Time (NOT1)	FactMdaCounter, DimMdaCounter, DimMdaOperation, DimWorkcenter	Sum of values multiplied by effective processing time, if performance is enabled and quality conditions are met.
Runtime	FactMdaState	Sum of the duration across all states.
Offtime	FactMdaState, DimMdaState, DimWorkcenter	Sum of duration where Kind is "Planned Stop" and availability is enabled.
Downtime	FactMdaState, DimMdaState, DimWorkcenter	Sum of duration where Kind is "Unplanned Stop" and availability is enabled.
Uptime	FactMdaState, DimMdaState, DimWorkcenter	Sum of duration where Kind is "Production" and availability is enabled.
Target Time (NOT2)	FactMdaState, DimMdaOperation, FactMdaMes, DimMdaState, DimWorkcenter	Sum of duration based on production or unplanned stop conditions when performance is not enabled.
Planned Production Time	FactMdaState	Calculated as Runtime minus Offtime.
Operating Time	FactMdaState	Calculated as PlanOccupancyTime minus Downtime.
Total Quantity	FactMdaCounter	Sum of GoodQuantity and ScrapQuantity.
Availability Rate	FactMdaMes	Calculated as AllUsageTime divided by PlanOccupancyTime.
Performance Efficiency	FactMdaMes	Calculated as sum of NOT1 and NOT2 divided by AllUsageTime.
Quality Rate	FactMdaMes	Calculated as (GoodQuantity - ReworkQuantity) divided by TotalQuantity.
Overall Equipment Effectiveness	FactMdaMes	Calculated as the product of Availability, Performance, and Quality.

A.5 Algorithm Implementation used by Package and Class

Algorithm	Python Package	Algorithm Class
Linear Regression	sklearn.linear_model	LinearRegression
Ridge Regression	sklearn.linear_model	Ridge
Polynomial Regression	sklearn.preprocessing	PolynomialFeatures
Decision Tree Regression	sklearn.tree	DecisionTreeRegressor
Random Forest Regression	sklearn.ensemble	RandomForestRegressor
Support Vector Regression	sklearn.svm	SVR
eXtreme Gradient Boosting	xgboost	XGBRegressor
CatBoost	catboost	CatBoostRegressor
Light Gradient-Boosting Machine	lightgbm	LGBMRegressor
Neural Network	torch	nn, optim, nn.MSELoss()

A.6 Optimized Hyperparameters per Model Target

Algorithm	Optimized Hyperparameter Set
Target OEE	
Ridge Regression	alpha: 3.9099
Polynomial Regression	poly_degree: 2
Decision Tree Regression	max_depth: 20, min_samples_split: 11, min_samples_leaf: 5
Random Forest	n_estimators: 412, max_depth: 10, min_samples_split: 7, min_samples_leaf: 2, bootstrap: True
SVR	C: 4.1971, epsilon: 0.0267, kernel: rbf
XGBoost	n_estimators: 186, max_depth: 5, learning_rate: 0.1471, subsample: 0.8369, colsample_bytree: 0.7563, gamma: 0.0140
CatBoost	iterations: 205, learning_rate: 0.0581, depth: 6, l2_leaf_reg: 9, border_count: 210, grow_policy: Depthwise
LightGBM	n_estimators: 238, learning_rate: 0.1731, num_leaves: 219, boosting_type: dart, colsample_bytree: 0.8898, reg_alpha: 0.5541, reg_lambda: 0.8333
Neural Network	num_epochs: 1035, learning_rate: 0.0052, early_stop_patience: 193
Target AR	
Ridge Regression	alpha: 2.5132
Polynomial Regression	poly_degree: 2
Decision Tree Regression	max_depth: 15, min_samples_split: 11, min_samples_leaf: 4
Random Forest	n_estimators: 207, max_depth: 13, min_samples_split: 7, min_samples_leaf: 2, bootstrap: True
SVR	C: 2.2576, epsilon: 0.0614, kernel: rbf
XGBoost	n_estimators: 261, max_depth: 4, learning_rate: 0.1474, subsample: 0.9980, colsample_bytree: 0.9980, gamma: 0.0090
CatBoost	iterations: 422, learning_rate: 0.0503, depth: 5, l2_leaf_reg: 4, border_count: 139, grow_policy: Lossguide
LightGBM	n_estimators: 415, learning_rate: 0.0637, num_leaves: 163, boosting_type: gbdt, colsample_bytree: 0.6816, reg_alpha: 0.9999, reg_lambda: 0.9972
Neural Network	num_epochs: 116, learning_rate: 0.0148, early_stop_patience: 70
Target PE	
Ridge Regression	alpha: 2.6029
Polynomial Regression	poly_degree: 2
Decision Tree Regression	max_depth: 33, min_samples_split: 11, min_samples_leaf: 5
Random Forest	n_estimators: 334, max_depth: 10, min_samples_split: 5, min_samples_leaf: 1, bootstrap: True
SVR	C: 2.6170, epsilon: 0.0344, kernel: rbf
XGBoost	n_estimators: 199, max_depth: 6, learning_rate: 0.0306, subsample: 0.7900, colsample_bytree: 0.8691, gamma: 0.0442
CatBoost	iterations: 209, learning_rate: 0.0568, depth: 10, l2_leaf_reg: 9, border_count: 127, grow_policy: Lossguide
LightGBM	n_estimators: 450, learning_rate: 0.1541, num_leaves: 235, boosting_type: dart, colsample_bytree: 0.7892, reg_alpha: 0.6987, reg_lambda: 0.1050
Neural Network	num_epochs: 160, learning_rate: 0.0346, early_stop_patience: 28
Target QR	

Continued on next page

Algorithm	Optimized Hyperparameter Set
Ridge Regression	alpha: 0.1258
Polynomial Regression	poly_degree: 2
Decision Tree Regression	max_depth: 38, min_samples_split: 5, min_samples_leaf: 5
Random Forest	n_estimators: 257, max_depth: 38, min_samples_split: 3, min_samples_leaf: 5, bootstrap: True
SVR	C: 0.3820, epsilon: 0.0109, kernel: rbf
XGBoost	n_estimators: 355, max_depth: 9, learning_rate: 0.1879, subsample: 0.7621, colsample_bytree: 0.6059, gamma: 0.0120
CatBoost	iterations: 495, learning_rate: 0.0125, depth: 8, l2_leaf_reg: 9, border_count: 164, grow_policy: Depthwise
LightGBM	n_estimators: 284, learning_rate: 0.0108, num_leaves: 153, boosting_type: goss, colsample_bytree: 0.7374, reg_alpha: 0.0110, reg_lambda: 0.0230
Neural Network	num_epochs: 1040, learning_rate: 0.0236, early_stop_patience: 202
Target PPT	
Ridge Regression	alpha: 6.7017
Polynomial Regression	poly_degree: 2
Decision Tree Regression	max_depth: 40, min_samples_split: 5, min_samples_leaf: 5
Random Forest	n_estimators: 314, max_depth: 10, min_samples_split: 4, min_samples_leaf: 5, bootstrap: True
SVR	C: 9.9587, epsilon: 0.4808, kernel: rbf
XGBoost	n_estimators: 426, max_depth: 3, learning_rate: 0.0213, subsample: 0.7605, colsample_bytree: 0.9647, gamma: 0.2939
CatBoost	iterations: 197, learning_rate: 0.0315, depth: 6, l2_leaf_reg: 7, border_count: 248, grow_policy: Depthwise
LightGBM	n_estimators: 443, learning_rate: 0.0136, num_leaves: 102, boosting_type: goss, colsample_bytree: 0.7435, reg_alpha: 0.8041, reg_lambda: 0.7949
Neural Network	num_epochs: 1278, learning_rate: 0.0300, early_stop_patience: 35
Target OT	
Ridge Regression	alpha: 2.4945
Polynomial Regression	poly_degree: 2
Decision Tree Regression	max_depth: 24, min_samples_split: 3, min_samples_leaf: 1
Random Forest	n_estimators: 308, max_depth: 49, min_samples_split: 8, min_samples_leaf: 1, bootstrap: True
SVR	C: 9.7129, epsilon: 0.0414, kernel: rbf
XGBoost	n_estimators: 106, max_depth: 5, learning_rate: 0.0504, subsample: 0.7380, colsample_bytree: 0.8921, gamma: 0.0057
CatBoost	iterations: 341, learning_rate: 0.0221, depth: 7, l2_leaf_reg: 3, border_count: 141, grow_policy: Depthwise
LightGBM	n_estimators: 380, learning_rate: 0.1440, num_leaves: 255, boosting_type: dart, colsample_bytree: 0.9863, reg_alpha: 0.3287, reg_lambda: 0.7239
Neural Network	num_epochs: 1724, learning_rate: 0.0989, early_stop_patience: 69
Target DT	
Ridge Regression	alpha: 14.6986
Polynomial Regression	poly_degree: 2

Continued on next page

Algorithm	Optimized Hyperparameter Set
Decision Tree Regression	max_depth: 10, min_samples_split: 11, min_samples_leaf: 5
Random Forest	n_estimators: 156, max_depth: 26, min_samples_split: 3, min_samples_leaf: 5, bootstrap: True
SVR	C: 0.5901, epsilon: 0.4489, kernel: linear
XGBoost	n_estimators: 182, max_depth: 4, learning_rate: 0.0133, subsample: 0.7075, colsample_bytree: 0.6104, gamma: 0.0003
CatBoost	iterations: 211, learning_rate: 0.0165, depth: 8, l2_leaf_reg: 6, border_count: 128, grow_policy: Lossguide
LightGBM	n_estimators: 51, learning_rate: 0.0614, num_leaves: 177, boosting_type: goss, colsample_bytree: 0.9605, reg_alpha: 0.7228, reg_lambda: 0.9876
Neural Network	num_epochs: 114, learning_rate: 0.0044, early_stop_patience: 242

Bibliography

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2623–2631, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701.

David M. Allen. Mean Square Error of Prediction as a Criterion for Selecting Variables. *Technometrics*, 13(3):469–475, August 1971. ISSN 0040-1706, 1537-2723. doi: 10.1080/00401706.1971.10488811. URL http://www.tandfonline.com/doi/abs/10.1080/00401706.1971.10488811.

Aayush Bajaj. Performance Metrics in Machine Learning [Complete Guide], July 2022. URL https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide.

James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of machine learning research*, 13(2):281–305, 2012.

Michael Braun. Nicht-funktionale Anforderungen, January 2016. URL https://www.pst.ifi.lmu.de/Lehre/wise-15-16/jur-pm/braun-ausarbeitung.pdf.

S. Chapman. The Fundamentals Of Production Planning And Control. March 2005. URL https://www.semanticscholar.org/paper/The-Fundamentals-Of-Production-Planning-And-Control-Chapman/554ff92970c557c80c78b337f01c56eecd1ddfea.

O. Chikwendu, Anozie Stephen Chima, and Mgbemena Chika Edith. The optimization of overall equipment effectiveness factors in a pharmaceutical company. *Heliyon*, 6, 2020. doi: 10.1016/j.heliyon.2020.e03796. URL https://consensus.app/papers/optimization-equipment-effectiveness-factors-company-chikwendu/6c5f20b0c2b959c68cf574851fef6e77/.

Péter Dobra and János Jósvai. Assembly Line Overall Equipment Effectiveness (OEE) Prediction from Human Estimation to Supervised Machine Learning. *Journal of Manufacturing and Materials Processing*, 6(3):59, June 2022a. ISSN 2504-4494. doi: 10.3390/jmmp6030059. URL https://www.mdpi.com/2504-4494/6/3/59. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

Péter Dobra and János Jósvai. Predicting the impact of type changes on Overall Equipment Effectiveness (OEE) through machine learning. In *2022 IEEE 1st International Conference on Internet of Digital Reality (IoD)*, pages 000011–000016, June 2022b. doi: 10.1109/IoD55468.2022.9986645. URL https://ieeexplore.ieee.org/abstract/document/9986645.

Choumicha El Mazgualdi, Tawfik Masrour, Ibtissam El Hassani, and Abdelmoula Khdoudi. Machine learning for KPIs prediction: a case study of the overall equipment effectiveness within the automotive industry. *Soft Computing*, 25 (4):2891–2909, February 2021. ISSN 1432-7643, 1433-7479. doi: 10.1007/s00500-020-05348-y. URL https://link.springer.com/10.1007/s00500-020-05348-y.

Adrien Payong Fadheli, Abdeladim. K-Fold Cross Validation using Scikit-Learn in Python - The Python Code, September 2024. URL https://thepythoncode.com/article/kfold-cross-validation-using-sklearn-in-python.

Flask. Welcome to Flask — Flask Documentation (3.0.x), September 2024. URL https://flask.palletsprojects.com/en/3.0.x/.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 196. MIT Press, Massachusetts, 2016. ISBN 978-0-262-03561-3. OCLC: 1091224171.

Michael Haenlein and Andreas Kaplan. A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. *California Management Review*, July 2019. doi: 10.1177/0008125619864925. URL https://journals.sagepub.com/doi/epub/10.1177/0008125619864925. Publisher: SAGE PublicationsSage CA: Los Angeles, CA.

Kai Holzweißig. *Wissenschaftliches Arbeiten - Eine Anleitung für dual Studierende der Wirtschaftsinformatik*. Leanpub, 4. auflage edition, April 2019. URL https://leanpub.com/wawinfo.

Vorne Industries. OEE Factors: Availability, Performance, and Quality | OEE, August 2024a. URL https://www.oee.com/oee-factors/.

Vorne Industries. OEE Calculation: Definitions, Formulas, and Examples | OEE, August 2024b. URL https://www.oee.com/calculating-oee/.

Vorne Industries. Six Big Losses in Manufacturing | OEE, August 2024c. URL https://www.oee.com/oee-six-big-losses/.

Vorne Industries. World-Class OEE: Set Targets To Drive Improvement | OEE, August 2024d. URL https://www.oee.com/world-class-oee/.

M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, July 2015. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aaa8415. URL https://www.science.org/doi/10.1126/science.aaa8415.

Yoonsuh Jung. Multiple predicting K -fold cross-validation for model selection. *Journal of Nonparametric Statistics*, 30(1):197–215, January 2018. ISSN 1048-5252, 1029-0311. doi: 10.1080/10485252.2017.1404598. URL https://www.tandfonline.com/doi/full/10.1080/10485252.2017.1404598.

Raman Kumar, Sita Rani, and Sehijpal Singh Khangura. *Machine Learning for Sustainable Manufacturing in Industry 4.0: Concept, Concerns and Applications*. CRC Press, November 2023. ISBN 978-1-00-098619-8. Google-Books-ID: vKMIEQAAQBAJ.

Laura Lucantoni, Sara Antomarioni, Filippo Emanuele Ciarapica, and Maurizio Bevilacqua. A rule-based machine learning methodology for the proactive improvement of OEE: a real case study. *International Journal of Quality & Reliability Management*, 41(5):1356–1376, January 2023. ISSN 0265-671X. doi: 10.1108/IJQRM-01-2023-0012. URL https://doi.org/10.1108/IJQRM-01-2023-0012. Publisher: Emerald Publishing Limited.

Luke Merrick and Ankur Taly. The Explanation Game: Explaining Machine Learning Models Using Shapley Values. volume 12279, pages 17–38, Cham, 2020. Springer International Publishing. ISBN 978-3-030-57320-1 978-3-030-57321-8. doi: 10.1007/978-3-030-57321-8_2. URL https://link.springer.com/10.1007/978-3-030-57321-8_2. Book Title: Machine Learning and Knowledge Extraction Series Title: Lecture Notes in Computer Science.

Ercan Oztemel and Samet Gursev. Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1):127–182, January 2020. ISSN 1572-8145. doi: 10.1007/s10845-018-1433-8. URL https://doi.org/10.1007/s10845-018-1433-8.

Andrew J. Patton and A. Timmermann. Testing Forecast Optimality Under Unknown Loss. *Journal of the American Statistical Association*, 102:1172–1184, 2007. doi: 10.1198/016214506000001176. URL https://consensus.app/papers/testing-forecast-optimality-under-unknown-loss-patton/b69e663bece052d89b82db1e7e41e91b/.

Dr. Pfleger. Dr. Pfleger Arzneimittel heute, 2024a. URL https://dr-pfleger.de/unternehmen/dr-pfleger-arzneimittel-heute/.

Dr. Pfleger. Wie wir arbeiten, 2024b. URL https://dr-pfleger.de/unternehmen/dr-pfleger-arzneimittel-heute/wie-wir-arbeiten/.

Liliane M-y A Pintelon and Peter Nganga Muchiri. Performance measurement using overall equipment effectiveness (OEE): literature review and practical application discussion. *International Journal of Production Research*, 46(13):3517–3535, July 2008. ISSN 0020-7543. doi: 10.1080/00207540601142645. URL https://doi.org/10.1080/00207540601142645. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207540601142645.

REFA. OEE - Overall Equipment Effectiveness, August 2024. URL https://refa.de/service/refa-lexikon/oee-overall-equipment-effectiveness.

Biswaranjan Senapati, Awad Bin Naeem, and Renato R. Maaliw. Machine Learning Model for Improving the Overall Equipment Effectiveness in Industrial Manufacturing Sites. In *Advances in Computational Intelligence and Its Applications*. CRC Press, 2024. ISBN 978-1-00-348868-2. Num Pages: 11.

Christian Stemper, Eva Kodisch, and David Billing. AI Usecases @ Dr. Pfleger - Optimization of Production Planning, November 2022.

Juan Pablo Usuga Cadavid, Samir Lamouri, Bernard Grabot, Robert Pellerin, and Arnaud Fortin. Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing*, 31(6):1531–1558, August 2020. ISSN 1572-8145. doi: 10.1007/s10845-019-01531-7. URL https://doi.org/10.1007/s10845-019-01531-7.

Bruno Vilela De Souza, Sérgio R. Barros Dos Santos, André M. De Oliveira, and Sidney N. Givigi. Analyzing and Predicting Overall Equipment Effectiveness in Manufacturing Industries using Machine Learning. In *2022 IEEE International Systems Conference (SysCon)*, pages 1–8, April 2022. doi: 10.1109/SysCon53536.2022.9773846. URL https://ieeexplore.ieee.org/document/9773846. ISSN: 2472-9647.

Thomas Wilde and Thomas Hess. Forschungsmethoden der Wirtschaftsinformatik. *WIRTSCHAFTSINFORMATIK*, 49(4):280–287, August 2007. ISSN 1861-8936. doi: 10.1007/s11576-007-0064-z. URL https://doi.org/10.1007/s11576-007-0064-z.

Tzu-Tsung Wong and Nai-Yu Yang. Dependency Analysis of Accuracy Estimates in k-Fold Cross Validation. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2417–2427, November 2017. ISSN 1041-4347. doi: 10.1109/TKDE.2017.2740926. URL http://ieeexplore.ieee.org/document/8012491/.

Carl August Zehnder. *Informationssysteme und Datenbanken*. Leitfäden der Informatik. Teubner Verlag, Stuttgart, 6. auflage edition, 1998. ISBN 3-519-32480-6.

Muhammad Zubair, S. Maqsood, Tufail Habib, Qazi Muhammad Usman Jan, Uroosa Nadir, M. Waseem, and Q. Yaseen. Manufacturing productivity analysis by applying overall equipment effectiveness metric in a pharmaceutical industry. *Cogent Engineering*, 8, 2021. doi: 10.1080/23311916.2021.1953681. URL https://consensus.app/papers/manufacturing-productivity-analysis-applying-equipment-zubair/98e794c62e3e542f85f9c5924bf02bf8/.

Declaration of Authorship

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Abschlussarbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Place, Date

Signature