FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Application of Machine Learning in Production Scheduling

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Georg Faustmann, BSc
Matrikelnummer 1326020

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Privatdoz. Dipl.-Ing. Dr.techn. Nysret Musliu

Wien, 8. Oktober 2019

Georg Faustmann	Nysret Musliu

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Application of Machine Learning in Production Scheduling

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Georg Faustmann, BSc
Registration Number 1326020

to the Faculty of Informatics

at the TU Wien

Advisor: Privatdoz. Dipl.-Ing. Dr.techn. Nysret Musliu

Vienna, 8th October, 2019

Georg Faustmann	Nysret Musliu

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Georg Faustmann, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. Oktober 2019

Georg Faustmann

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I am deeply grateful for the possibility to do my Master's thesis in cooperation with the Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling under the supervision of Priv.-Doz. Dr. Nysret Musliu. He always had a sympathetic ear about issues related to the thesis and found a good tradeoff between realizing own ideas and steering the project. It was a great experience to work with him.

I also want to thank Felix Winter MSc for his great ability to analyze problems in a structured way which often helps to improve solutions. He also proof-read a part of the thesis.

A Christian Doppler Laboratory tries to combine basic research with the industry. I say thank you to the Christian Doppler Research Society for the opportunity. I like the idea to combine these two worlds. My industry partner was MCP which was a inspiring place for me. Especially I want to thank Christoph Mrkvicka MSc who was my main contact at MCP. It was always possible to schedule an appointment with him to discussed different solution methods.

I would also like to thank my colleague Maximilian Moser which was always a good contact to discuss critically various solution approaches.

I acknowledge that Johannes Gösweiner and Christopher Kawczynski proof-read the thesis.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Für menschliche Experten ist es oft schwierig oder zeitaufwendig, Muster aus großen Datenmengen manuell zu erfassen. Maschinelles Lernen wird in vielen Gebieten eingesetzt, um solche Muster computergestützt zu erkennen. Seine Anwendungen sind keineswegs nur auf Forschung reduziert, sondern spielt auch in der Industrie eine große Rolle. Diese Arbeit verwendet maschinelles Lernen für zwei wirtschaftlich relevante Themen. Der erste Teil der Arbeit handelt von der Vorhersage der Qualität von herzustellenden Produkten in der Automobilindustrie. Der zweite Teil untersucht einen Ansatz, der automatisiert Parametereinstellen für Planungsregeln wählt. Diese Regeln finden in der Ablaufplanung für Maschinen Anwendung.

Um in der Industrie Kosten und Produktionszeit zu senken, gibt es großes Interesse an Möglichkeiten, die Produktqualität bereits vor dem Herstellungsprozess abzuschätzen. Wir verwenden binäre Klassifizierung, um die Qualität von Produkten aus Lackieranlagen der Automobilindustrie basierend auf Planungsdaten vorherzusagen. Dazu stellen wir mehrere Features vor, die den Produktzustand charakterisieren. Basierend auf diesen Features wird klassifiziert, ob die Produktqualität zufriedenstellend oder mangelhaft sein wird.

Eine große Auswahl an modernen maschinelle Lernmethoden wurden angewandt und analysiert. Zusätzlich haben wir automatisierte maschinelle Lerntechniken verwendet, um Klassifikatoren von hoher Güte zu erhalten. Wir können zeigen, dass das Beste von uns gefundene Modell bei Testdaten, welche das Modell zuvor noch nicht gesehen hat, eine bessere Leistung hat als ein Referenzmodell.

Das Finden von optimierter Sequenzplanung für Maschinen ist eine wichtige Aufgabe, welche in verschiedenen Branchen der Industrie auftaucht. In der Praxis werden oft Planungsregeln für das automatisierte Erstellen von Sequenzplanungen verwendet. Diese Regeln sind flexibel einsetzbar und liefern schon nach kurzer Laufzeit gute Ergebnisse. Manche Planungsregeln haben Parameter, welche einen großen Einfluss auf die Lösungsqualität einer gegebenen Planungsproblem-Instanz haben. Bei einem neuen Planungsszenario stellt momentan normalerweise ein menschlicher Experte diese Parameter manuell ein.

Im Teil Automatisches Parameter-Einstellen dieser Arbeit untersuchten wir maschinelle Lernmethoden, um unter realen Bedingungen automatisch Parameter von Planungsregeln unter Berücksichtigung mehrerer Zielvorgaben an den resultierenden Plan zu

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

konfigurieren. Diese maschinellen Lernmethoden basieren auf Regressionsmodellen, welche mehrere kontinuierliche Werte vorhersagen. Wir schlagen eine Menge von neuartigen Features vor, um Instanzen des Problems Sequenzplanung mit parallel laufenden Maschinen zu charakterisieren. Weiters beschreiben wir, wie überwachtes Lernen eingesetzt werden kann, um optimierte Parameter-Konfigurationen für eine gegebene Instanz eines Sequenzplanung-Problems für Maschinen zu erhalten. Experimentelle Ergebnisse zeigen, dass unser Ansatz es ermöglicht, in kurzer Zeit hochqualitative Pläne für reale Planungsszenarien zu erhalten.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Abstract

For human experts, it is often too hard or too time-consuming to manually detect patterns in big data sets. Machine learning is applied in many areas to detect such patterns. Its applications are by no means limited to research, as machine learning also plays a big role in the industrial sector. This thesis applies machine learning to the following two topics. The first part of the thesis deals with product quality classification for automotive paint shops. The second part investigates automated parameter configuration for dispatching rules that are used in machine scheduling.

To reduce costs and production time, estimating the production quality is important. We use a binary classification to predict the product quality of an automotive paint shop based on its scheduling data. We propose a set of features to characterize the production process. These features are used to classify whether or not the quality of the product is satisfactory.

A wide variety of state-of-the-art machine learning methods are applied and analyzed. Additionally, we consider an automated machine learning technique to obtain high-quality classifiers. We can show that the best model we found performs better than a baseline model on an unseen data set.

Finding optimized machine schedules is a very important task that arises in many areas of industrial manufacturing. In practice, dispatching rules are often suggested for automated creation of schedules, as such rules are flexible and able to provide good results within short run times. Many dispatching rules have parameters which can drastically affect the solution quality for a given instance. Currently, tuning the parameter configurations is usually performed by a human expert whenever a new planning scenario is encountered.

In the automated parameter configuration part of the thesis, we investigate machine learning methods based on multi-target regression to automatically configure dispatching rules for real-life planning scenarios where multiple objectives are considered. We propose a novel set of features to characterize instances of the parallel machine scheduling problem, and describe how supervised learning can be used to obtain optimized parameter configurations for given machine scheduling instances. Experimental results show that our approach can obtain high-quality solutions for real-life scheduling scenarios in short run times.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

In the industry, the amount of data gathered during production is increasing. It is time consuming to examine and draw conclusions from this data manually. Machine learning is a promising technique to get insights into the data in a computer-aided way. This thesis applies machine learning to two kinds of problems in production scheduling. The first one is about predicting product quality and the second one is about the automated parameter configuration for a heuristic.

The first part of the thesis proposes a machine learning approach to predict the product quality for an automotive paint shop. For example, Bai et al. [3] point out that the Chinese government has a heavily interested to make its industry ready for the future. This task which is referred as *industry 4.0* is not only limited to China. This goal includes the possibility to predict the manufacturing quality. This topic has been investigated within various industries. For example Chen et al. [15] use a neural network-based approach to predict the quality of plastic injection molding. Lange [34] applies machine learning to predict the paint thickness of the target geometry mesh. This helps to identify problematic surfaces on the product during painting. Ju et al. [32] use a manually created probabilistic model to estimate product quality after multiple operations.

Our approach aims to predict product quality based on the production schedule with machine learning. To train the machine learning models actual scheduling data is used from the automotive industry. We define features to characterize the state in which a product is lacquered and investigate different machine learning techniques including different data preparation steps. Further, we apply state-of-the-art techniques that help us in obtaining good performing classifiers and understanding the prediction of complex models. We evaluate the best-obtained classifier on unseen test data and show that the performance is superior to a baseline classifier.

The second part of the thesis deals with automated configuration of dispatching rules. Dispatching rules have been successfully applied to many scheduling problems including

parallel machine scheduling, which is a very important real-life problem. Mokotoff and Ethel [40] give an overview of many variants of this problem and shows that even the basic parallel machine scheduling problem is NP-hard. A machine scheduling problem instance consists of multiple jobs and multiple machines. The aim is to assign each job to a machine such that a number of constraints are fulfilled and they are are optimized according to multiple different objective functions.

Many dispatching rules have been proposed in the literature to obtain solutions for machine scheduling problems. For example, Panwalkar and Iskander [44] is a survey about such rules. Montazeri and Wassenhove [41] and Sabuncuoglu [49] conclude that no dispatching rule outperforms all other rules on every problem variant. Many dispatching rules can be configured via a number of parameters, and usually, the selected parameter configuration has a large impact on the quality of the generated solution. Their performance depends on the characteristics of the problem instances and objectives that should be minimized. Although in many practical applications the parameter configuration is still done manually by human-experts, several automated approaches have been proposed to automatically generate and configure dispatching rules.

Branke et al. [7] give an overview of hyper-heuristics that are used to find good dispatching rules. These rules are tuned to perform well on a given instance set. Heger et al. [27] and Mouelhi-Chibani and Pierreval [43] improve the dispatching-rule-based scheduling process by switching between different dispatching rules dynamically depending on the current schedule. In Burke et al. [12] a general overview of hyper-heuristics is given and it describes their application for automated selection of good dispatching rules.

In this thesis, we investigate an instance-based automated configuration of dispatching rule parameters. We introduce novel features to characterize parallel machine scheduling instances, that can be used to automatically obtain efficient dispatching rule parameters via machine learning. Dynamic dispatching rules that incorporate machine learning techniques (e.g. Heger et al. [27]) and problem independent automated algorithm configuration approaches (e.g. Hutter et al. [30]) have been applied in the past. This thesis analyses the automated configuration of the parameters with multi-target regression based on instance-specific features for parallel machine scheduling problems. As we propose a supervised machine learning approach, we further describe how a set of given parallel machine scheduling instances can be used to generate a training set. Additionally, we introduce a grid search approach that systematically evaluates the performance of the given dispatching rule over a given discrete parameter space. This grid search can be utilized to select appropriate ground truth values for our training set.

We apply the proposed method to automatically select parameters of a dispatching rule that is commonly used in the industry. This dispatching rule prioritizes jobs according to multiple attributes and is parameterized by a number of weights that configure the focus on the different attributes.

We evaluate the performance of the used machine learning models on unseen benchmark instances that are based on actual planning scenarios and compare the results to results

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

produced by a state-of-the-art automated algorithm configuration approach. The experiments show that our approach is able to determine parameter configurations that lead to high-quality solutions for practical parallel machine scheduling instances. Furthermore, the machine learning approach requires much less run time to obtain good parameter configurations for the benchmark instances compared to a state-of-the-art automated algorithm configuration method.

The following are the main contributions of the automated parameter configuration part of the thesis:

- We introduce a set of novel features to characterize parallel machine scheduling instances.
- We propose a supervised machine learning approach using multi-target regression to automatically configure dispatching rules for each specific instance.
- We compare the machine learning approach to a state-of-the-art automated algorithm configurator on a large set of instances and show the viability of our approach.

1.1 Structure of the thesis

Chapter 2 describes our approach for product quality prediction for an automotive paint shop. Chapter 3 deals with automated parameter configuration for a dispatching rule. In the beginning of both chapters, a brief description of the respective topics is given.

Concluding remarks and ideas for future work are given in Chapter 4.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Quality Prediction for Automotive Paint Shops

This chapter is about product quality prediction for an automotive paint shop. Section 2.1 describes how the considered paint shop works and explains why machine learning is used to predict the product quality. Section 2.2 gives an overview of current state-of-the-art product quality prediction methods and emerging research fields that are relevant to the machine learning task. Afterwards, in Section 2.3 we propose our approach to predict product quality for a paint shop and we briefly describe the investigated, well-researched, machine learning methods. In Section 2.4 we give experimental results and an evaluation of them.

2.1 Motivation and Problem Statement

For the paint industry, it is important to reduce costs and production time. One possibility to achieve this is by improving the production quality. For the industry, it is beneficial to estimate the quality based on the planned production condition beforehand.

This thesis proposes an approach to predict the quality of painted pieces of an automated paint shop. The produced painted pieces, such as bumpers and other exterior systems, are used for car manufacturing. The paint shop under consideration, lacquers multiple parts at once and these parts are attached to a skid. Multiple skids are conveyed through a production line which contains all the necessary stages for the paint shop process. Each skid must run through all stages, i.e. no stage can be bypassed. Each painting stage applies one coating in the following order: ground coating, base coating, and clear coating. A coating is a mixture of lacquer components. After the last stage of the lacquer process, a quality check of the coated pieces is performed where the outcome is either accepted or not. If one skid has to be coated with other lacquer components than the previous

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

one then it is necessary to change the lacquer accordingly. Some lacquer changes are known empirically to be problematic. For example after a change from red to white, red particles are likely to appear on the white-coated pieces.

The number of changes for one coating stage is approximately the number of its coatings to the power of two. In our investigated paint shop, the number of these combinations is greater than one hundred. Since every piece is painted in three stages, the number of combinations for the whole paint shop becomes too large to be analyzed manually. Additionally, there can also be other factors that lead to coated pieces of inferior quality. The paint shop has a high degree of automation, it is therefore unlikely that unpredictable errors by humans occur. This makes the whole process more deterministic.

This thesis uses machine learning to predict the quality outcome based solely on the scheduling plan. The generation of such a schedule is researched by Winter et al. [55]. Further more, this paper contains a detailed description of the paint shop scheduling problem.

The scheduling plan contains the skid type for each skid position, the attached pieces, and the applied coating combination. Consequently, no additional sensors providing detailed information concerning the production process needs to be installed and existing real-world schedules from the past can be used to train the machine learning model. The quality outcome is predicted as either okay or not okay, as it is done by the final quality check of the paint shop. Since only prior known data are considered, such a model can be used to rate multiple scheduling plans and prioritize the plan in which the predicted number of lacquered products of inferior quality is the least.

2.2 State Of the Art

Machine learning has been succesfully applied in a variety of different fields. Wuest et al. [56] summarize the advantages, challenges, and applications of machine learning in manufacturing. They describe the successful usage of supervised machine learning to predict the quality of different products. For example, Chen et al. [15] predict the quality of a plastic injection molding process with a neural network-based approach. Data about the process and the used molding machine are used as input for the model. Ribeiro [47] experiments with a support vector machine for the same problem. He uses a series of discrete values from sensor readings as the input. Bai et al. [3] mention the Chinese government initiated strategy "Made in China 2025" where the prediction of manufacturing quality plays a large role. Bai et al. [3] investigate different machine learning approaches to tackle this issue with manufacturing data sets. These works demonstrate successful applications of machine learning to predict product quality.

Machine learning has not yet been applied to predict the final quality of paint shop products based on scheduling data. However, other techniques have been investigated to estimate properties about paint shop processes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Ju et al. [32] estimate the quality of the unfinished parts after multiple operations of an automotive paint shop with a manually created probabilistic model called the three-state quality flow model. After each operation the model distinguishes between three states for a part. Namely, a good state without repair, a good state after repair and a defective state. The idea behind these states is that parts which have never been repaired will have different probabilities to pass inspection compared to those which have been repaired in previous operations.

Lange [34] applies machine learning to predict the paint thickness of the geometry mesh. This is helpful to identify problematic surfaces on the product during painting.

Automated machine learning is an emerging research field that aims to automatically select and tune machine learning algorithms based on the provided data set. Recent full automatic machine learining competitions show that auto-sklearn [19] is a strong performing method. This framework won six out of ten tracks in a competition from 2016 [25]. In a competition in 2018 [20], a slightly modified version of auto-sklearn was able to outperform all other 41 participating systems.

While the performance of machine learning models increases, the high-performing models tend to get harder to understand. According to Baehrens et al. [2] it is hard to explain *why* such a complex model gives a particular prediction for an individual instance.

In recent times, research was done to get a better understanding of complex machine learning models. One promising method is called SHAP from Lundberg and Lee [38]. In our case, this can be helpful to find the properties which lead to degraded lacquer quality.

2.3 Quality Prediction for Paint Shop Scheduling

The quality prediction part of the thesis describes the workflow to predict the product quality of an existing paint shop via supervised machine learning.

2.3.1 Analyzing and Extracting Raw Data

The labeled data set is extracted from the real-life database. The automotive industry provides raw data in the form of an SQL database backup. This database is used for the production system. It contains data about the paint shop schedule and information about already processed schedules. The backup contains a stage, in which the automated process is not completely established. Due to testing the paint shop process, data was modified manually in the SQL database. This sometimes leads to missing or inconsistent data. It is required to identify the data that is consistent and available before a schedule is processed.

Furthermore, we have to decide which granularity the predictions should have. Our approach predicts the quality of the painted parts that belong to one skid run. This results in 17942 skid runs in which the lacquer quality of $\approx 77\%$ are okay and $\approx 23\%$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

are not okay. The target variable is either *nok* meaning the end painting quality is unacceptable or *ok* if the painting quality is acceptable.

Defined Features

From the backup two feature sets are extracted. We call the simpler feature set *basic* and the other one *extended*. The basic feature set is a subset of the extended one. In our case, one skid run represents one machine learning instance. The following section defines two feature sets to characterize one skid run.

Basic Feature Set We first propose a basic feature set that contains elementary information.

Features about the current skid:

InsertFlag: 1 means the used skid is inserted newly in the current round. Otherwise, it is 0.

RemovalFlag: 1 means the used skid is removed after the current round. Otherwise, it is 0.

MaxPRChangingTime: The maximal required cleaning time for a primer lacquer change in seconds.

MaxBCChangingTime: This feature is the maximal required cleaning time for a base coating lacquer change in seconds.

MaxCCChangingTime: This feature is the maximal required cleaning time for a clear coating lacquer change in seconds.

num_of_diff_parts: The number of different part types that are mounted on the current skid.

ProcessTime: The needed time to lacquer the parts on the current skid.

Previous skid features To characterize the state of the current skid run we extract the previously mentioned features for the three preceding skid runs.

The design of the given database groups multiple consecutive skids runs into rounds. If the skid of the current skid run is also used in the last two previous rounds the features mentioned beforehand are extracted from these previous skid runs. If the current skid is not used in a previous round then default values are set instead. For integer, float, and timestamps the default value is 0. The default string value is *None*.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Aggregating features These features aggregate states of preceding skid runs.

num_of_inserted_skids_current_round: It defines the number of inserted skids in the current round up to the current skid position.

num_of_removed_skids_previous_round: It defines the number of removed skids in the previous round.

num_of_skids_current_round: It defines the number of used skids in the current round.

num_of_skids_previous_round: It defines the number of used skids in the previous round.

number_of_previous_skids_with_same_pr_lacquer: It defines how many previous skid runs use the current primer coating without an interruption.

number_of_previous_skids_with_same_bc_lacquer: It defines how many previous skid runs use the current base coating without an interruption.

number_of_previous_skids_with_same_cc_lacquer: It defines how many previous skid runs use the current clear coating without an interruption.

different_part_combinations_in_sliding_window: It defines how many different part combinations are used within a defined window of previous skids.

num_of_inserted_skids_in_sliding_window: It defines how many skid insertions happened within a defined window of previous skids.

num_of_removed_skids_in_sliding_window: It defines how many skids are marked for removal within a defined window of previous skids.

counted_rounds_skid_used_without_removal: It defines how often the current skid was used in previous rounds.

different_colors_in_sliding_window: It defines how many different lacquer combinations are used within a defined sliding window of previous skids. This considers all three lacquer coatings.

different_cc_in_sliding_window: It defines how many different clear coating lacquer combinations are used within a defined sliding window of previous skids.

different_bc_in_sliding_window: It defines how many different base coating lacquer combinations are used within a defined sliding window of previous skids.

different_pr_in_sliding_window: It defines how many different primer lacquer combinations are used within a defined sliding window of previous skids.

different_all_in_sliding_window: It defines how often all lacquer coatings change between two consecutive skids within a defined sliding window of previous skids.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

are_different_colors_in_sliding_window: This feature is 1 if different_colors_in_sliding_window is not equal to 0. Otherwise, it is 0.

are_different_cc_in_sliding_window: This feature is 1 if different_cc_in_sliding_window is not equal to 0. Otherwise, it is 0.

are_different_bc_in_sliding_window: This feature is 1 if different_bc_in_sliding_window is not equal to 0. Otherwise, it is 0.

are_different_pr_in_sliding_window: This feature is 1 if different_pr_in_sliding_window is not equal to 0. Otherwise, it is 0.

are_different_all_in_sliding_window: This feature is 1 if different_all_in_sliding_window is not equal to 0. Otherwise, it is 0.

different_color_to_previous_position: This feature is 1 if any lacquer coating changes between the current and the previous skid. Otherwise, it is 0.

different_bc_to_previous_position: This feature is 1 if the base coating changes between the current and the previous skid. Otherwise, the feature value is 0.

different_cc_to_previous_position: This feature is 1 if the clear coating changes between the current and the previous skid. Otherwise, the feature value is 0.

different_pr_to_previous_position: This feature is 1 if the primer coating changes between the current and the previous skid. Otherwise, the feature value is 0.

deltaPosition: It defines the number of preceding skids runs where no parts are attached to it.

Extended Feature Set It extends the basic feature set with details about the lacquer components, which parts are attached to the skid and the skid type:

LacqBC: The used lacquer components of the base coating of the current skid run.

LacqPR: The used lacquer components of the primer coating of the current skid run.

LacqCC: The used lacquer components of the clear coating of the current skid run.

MatPerPos: Unique identifications for the part types which are attached on the skid of the current run.

SkidTypeId: The type of the used skid by a string representation.

The values of these features are categorical strings. The different lacquer coatings can contain multiple components. These components are all listed and separated with commas.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The same property holds for the part types. Parts of different types can be attached to one skid. All these types are listed and separated with commas.

The features are categorical strings and therefore they cannot be directly fed into a machine learning model. Section 2.3.2 describes how we tackled this issue.

These features are gathered for all the considered skids within the basic feature set. This includes the current skid at position p in round r, the previous skids at position $p-1$, $p-2$ and $p-3$ and the same skid used in the round $r-1$ and $r-2$. If there is no skid run at one desired position then default values are used for all the features as explained in the previous paragraph about the basic feature set.

Optimize Sliding Windows

Some features depend on the aggregation of previous skids runs. Thus the question arises: How many of such skid runs should be considered such that the most useful information is extracted? We group the considered previous skids to a *sliding window*.

The sliding window is optimized for each aggregated feature separately.

In general, the ideal feature separates the instances of the machine learning data set into pure partitions. This means that every instance with the same feature value belongs to the same target class.

The concept of information gain from Shannon [50] is used to indicate how well a feature separates the machine learning data set.

This concept quantifies in terms of entropy. The entropy for a binary class set is defined as:

$$Entropy\ H(binarySet) = -\frac{m}{m+n} * \log_2(\frac{m}{m+n}) - \frac{n}{m+n} * \log_2(\frac{n}{m+n})$$

In our case, n is the number of instances belonging to the target class *ok* and m is the number of instances belonging to the target class *nok*.

The Information Gain is defined as

$$Gain(InstanceSet,Feature) = H(InstanceSet) - \sum_{v \in Values(Feature)} \frac{|InstanceSet_v|}{|InstanceSet|} * H(InstanceSet_v)$$

, where H is the entropy.

For each feature, the sliding window with the highest information gain is obtained within the window size between $[2, 20]$.

Results of the optimized window sizes The optimized features with their best sliding window size are:

- different_colors_in_sliding_window: 17

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- different_bc_in_sliding_window: 18
- different_pr_in_sliding_window: 6
- different_cc_in_sliding_window: 6
- different_all_in_sliding_window: 5
- are_different_colors_in_sliding_window: 17
- are_different_bc_in_sliding_window: 5
- are_different_pr_in_sliding_window: 5
- are_different_cc_in_sliding_window: 4
- are_different_all_in_sliding_window: 6

2.3.2 Data Preparation

One-hot Encoding for String Features

The extended data set contains features of type string about the lacquer components, which parts are attached to a skid and the used skid type. The different lacquer coatings can contain multiple components and the parts that are mounted on the skid can be of different types.

These features are one-hot encoded in such a way that for every lacquer coating component, every part type and every skid type a feature is defined. For every component and part type that appears in an instance, the corresponding feature is set to 1. Otherwise the feature is set to 0. This means that one such string feature value can be transformed into a one-hot encoded feature set where multiple features are set to 1. For example, if the feature about the currently used base coating components *LacqBC* has the value "R43364,R43816,R44088", then the features *LacqBC_R43364*, *LacqBC_R43816* and *LacqBC_R44088* are set to 1 and all the other base coating component related features are set to 0. The skid type feature is transformed into a one-hot encoded feature set where exactly one feature is set to 1.

Unbalanced Data Set

In some machine learning data sets, the distribution of the target classes is not even. Our obtained dataset is unbalanced because more than 75% of the instances belong to the target class *ok* and the remaining instances belong to the target class *nok*. This is because the paint shop tries to avoid inferior paint quality. The problem that arises is that the machine learning model gets biased towards predicting the target class *ok*. López et al. [36] emphasize the importance of taking care of that problem and ways how to circumvent it.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

We investigated the two techniques named under-sampling and up-sampling to balance the dataset.

Random Under-sampling Random under-sampling creates a balanced dataset by matching the number of samples in the minority class with a random sample from the majority class.

Up-sampling Up-sampling matches the number of samples in the majority class with a resampling from the minority class. The most common technique is known as the Synthetic Minority Oversampling Technique.

Synthetic Minority Over-sampling Technique (SMOTE) This concept was introduced by Chawla et al. [14]. To illustrate this technique, consider some training data which has s samples, and f features in the feature space of the data. Note that these features, for simplicity, are continuous. To over-sample, take a random instance from the minority class and consider its k nearest neighbors in feature space. The synthetic instance for the minority class is generated by taking the vector between one of those k neighbors, and the randomly selected instance. This vector is multiplied by a random number between $[0, 1]$. The higher the random number is, the higher the similarity of the synthetic instance to the selected neighbor is and vice-versa.

It is important to not include test instances for the machine learning model in the up-sampling process because otherwise, the newly generated synthetic instances can contain information about them.

Combination of Under-sampling and Up-Sampling: SMOTE showed that it generates noisy samples by interpolating new points between marginal outliers and inliers. This issue can be solved by filtering the space resulting from the over-sampling method SMOTE.

Tomek link is a filtering method that is explained in Kotsiantis et al. [33]. A Tomek link is between instances that are each other's closest neighbors in the feature space but do not share the same class label. With this identified relationship both instances are removed. The process repeats until no more Tomek links can be found.

Scale Feature Values

Many machine learning algorithms work better when features are close to a normal distribution and are on a relatively similar scale.

The algorithms might increase the performance or converge faster. For example, for k-nearest neighbor it is helpful to have a similar range of features because the prediction is based on the distance between feature values. Hence it is vital to normalize the feature values so that each feature gets a similar relevance for the prediction.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Standard Normal Distribution Scaler This scaler subtracts the mean from each feature value based on all feature values to center the resulting mean value to 0. Then the values are scaled by dividing by their standard deviation such that the resulting standard deviation becomes 1. The resulting feature values imitate a standard normal distribution.

Range Scaler This technique translates each feature individually such that the values are within a defined range, for example between zero and one.

The following constant values are used for the scaling:

- x_{min}: minimal feature value in the test set
- x_{max}: maximal feature value in the test set
- new_{min}: starting point of the scaled range
- new_{max}: end point of the scaled range

A feature value x is scaled according to Equation (2.1).

$$x_{new}(x) = \frac{x - x_{min}}{x_{max} - x_{min}} * (new_{max} - new_{min}) + new_{min} \tag{2.1}$$

Feature Selection

If too many features are considered, an effect called the curse of dimensionality may occur. This effect is analyzed by Trunk [53]. The curse of dimensionality happens if the number of possible feature value combinations increases drastically with the number of features. Therefore, a training set that is too small results in a sparsely populated feature space. That makes it hard for a machine learning algorithm to extract patterns based on the training set.

Feature selection reduces the feature dimensionality to improve estimators' accuracy scores and also to boost their run time performance.

Select K Best The k highest scoring features based on a scoring function are kept. For example, the statistical concept ANOVA F-value can be used as a scoring function.

Feature Dimensionality Reduction

Feature projection transforms data from a high-dimensional space to a lower-dimensional space. In contrast to feature selection, feature dimensionality reduction tries to convey the information of all features into a lower-dimensional space.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Principal Component Analysis Principal Component Analysis (PCA), analyzed in Jolliffe [31], is a linear data transformation. PCA projects the features onto the principal components. The first principal component is the line that maximizes the variance of the projected values along this line. The second greatest variance on the second component, and so on.

2.3.3 Binary Classification

Classification is a machine learning task where a model predicts an element from a finite class set based on given feature values as a target. Such models are trained in a supervised fashion and thus require a labeled data set, meaning a data set with feature values and the corresponding target class. Our problem is tackled with binary classification. That means a classification where the target class set is of cardinality 2.

Classification Algorithms

The following gives an overview and a short description of the used classification methods.

Support Vector Machine For the binary classification task the support vector machine method (see: Chang and Lin [13]) finds a hyperplane that separates the instances of the two classes in the feature space. This hyperplane can be seen as a decision boundary. The points which are closest to that hyperplane are called support vectors. The distance from the hyperplane to these support vectors is called a margin. The support vector machine method maximizes the margin. This pure method is only useful for linearly separable data sets. Thus kernels are used to convert features into higher dimensions such that they become linearly separable.

k-nearest neighbors This technique is simple yet often effective in practice. Goldberger et al. [24] successfully applied this technique and explained the details about k-nearest neighbors. The prediction of this technique is based on the k instances from the training set that are near the considered instance in the feature space.

Decision Tree A decision tree based on Breiman et al. [11] is a directed tree in which internal nodes perform a check on a feature (For example, identifying which interval a feature value belongs to). Each branch of the node represents the outcome of the check (For example the different intervals). The leaf nodes represent class labels. Classification is done by starting at the root node and the predicted class is obtained at the bottom of the tree at a leaf node. This method has the advantage that the traced path from the root node to the leaf node *explains* why the decision tree comes to a certain prediction.

Multilayer perceptron This classifier is inspired by the human brain. It contains layers of perceptrons. The first layer of a multilayer perceptron is the input layer. It is followed by hidden layers. Each hidden layer is fully connected with the perceptrons

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

of the previous layer. The last layer is fully connected with the last hidden layer and is called the output layer. Each perceptron performs only a simple computation. It calculates a weighted sum of the output of the perceptrons of the previous layers and adds a weighted bias value. The output of a perceptron is the result of an activation function applied to this sum. The multilayer perceptron learns patterns from a data set by adjusting the weight values of the perceptrons. Hinton [28] gives an in-depth insight on how this technique works.

Naive Bayes Classifier The base of this classifier is the Bayes Theorem.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

A and B represent events. For example A could be that the target class is c out of the possible class set C (more formally: $C = c$) and B could be that the n^{th} feature is equal to f_n. Due to readability, the event is simply denoted with c respectively f_n. Naive Bayes assumes independence between the features.

It calculates for each target class c the probability given the feature vector $F = (f_1, f_2, \ldots, f_n)$.

$$P(c|f_1, f_2, \ldots, f_n) = \frac{P(f_1|c)P(f_2|c)P(f_2|c)\ldots P(f_n|c)P(c)}{P(f_1, f_2, \ldots, f_n)}$$

Since the denominator is constant for all the target classes c it can be omitted for the following calculations. The target class t that maximizes the probability that t occurs given the feature vector F is predicted.

$$t = \text{argmax}_c P(c)\Pi_{i=1}^{n} P(f_i|c)$$

Random Forest A random forest based on Breiman [10] is a collection of decision trees. The driving principle is to build several estimators independently and to average their predictions. On average, the combined estimator is usually better than any of the single base estimators because its variance is reduced. These trees are not trained on the entire data set with all features as it is done for a normal decision tree classifier. This method picks for each tree a random subset of features and a random subset of training instances. The prediction of a random forest is based on a voting from all decision trees. For classification, a majority voting is often used.

AdaBoost Freund and Schapire [21] describe this technique that trains a set of weak classifiers and combines them into one strong classifier. For each weak classifier, a weight based on its performance is determined. An AdaBoost classifier is a meta-estimator that starts by training a weak classifier on the original data set and then trains further weak classifiers on the same data set, but with the focus on incorrectly classified instances such that subsequent classifiers take more care on difficult cases. The more test instances

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

	actual okay quality (ok)	**actual inferior quality (nok)**
predicted okay quality (ok)	0	$C_{ok,nok}$
predicted inferior quality (nok)	$C_{nok,ok}$	0

Table 2.1: Cost matrix

a weak classifier predicts correctly the higher its weight for prediction. In our case, a decision tree with one decision node is used as weak classifier.

Gradient Boosting During the learning process, the gradient boosting method sequentially builds models. The first model is trained on a subset of the data set to predict the target class. Afterward, the loss, meaning the difference between the outcome of the classifier and the ground truth value, is calculated. The second model is trained to predict the loss of the first model. The third is trained to predict the loss of the second model and so forth until the desired number of models is reached. Friedman [22] gives a detailed description of this method.

Bagging Classifier The two main differences compared to the random forest method are the following: The bagging method described in Breiman [9] considers all features for to obtain the best split at a node, while the random forest method uses only a random subset of features. The second difference is that the base classifiers are not restricted to decision trees.

Extra Trees The differences of the extra trees method described in Geurts et al. [23] compared to the random forest method are:

1. When choosing features at a split, samples are drawn from the entire training set instead of a bootstrap sample of the training set.
2. At every node a random split among a random subset of the features is selected.

Cost Sensitive Classifier

A cost sensitive classifier as described in Elkan [18] takes the cost of every error type into account to avoid costly predictions. The costs of the different error types are represented by a cost matrix C. The matrix entry C_{ij} is the cost of predicting the i^{th} label while the j^{th} label is the correct label. In general predicting the right label has lower cost than faulty prediction, $\forall i \neq j : C_{ij} > C_{ii}$. For our binary classification the matrix looks like Table 2.1 where the cost for a correct prediction is 0.

We used the proposed function from Elkan [18] to select the optimal decision which take misclassification costs into account. For the instance x the optimal label is the i^{th} label which minimizes the cost function:

$$L(x, i) = \sum_{j \in Labels} P(j|x) \cdot C_{ij}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

For an instance x the cost to predict the label i is determined by a sum where the summands are the classification probabilities from the machine learning model for all classes $j \in Labels$ multiplied with the cost C_{ij}.

Automated Machine Learning

Automated machine learning is an emerging research field. It investigates algorithms to build machine learning models with no human intervention.

Auto-sklearn Auto-sklearn from Feurer et al. [19] frees the machine learning user from data preparation steps, algorithm selection, and hyperparameter tuning. It leverages recent advantages in Bayesian optimization, meta-learning and ensemble construction.

Auto-sklearn uses Bayesian optimization to tune hyperparameters. In short, Bayesian optimization builds a model itself to predict the performance of the hyperparameters. Based on this model it is decided which hyperparameter configuration should be selected next. The selection process is a trade-off between exploitation that selects configurations where the model expects good results and exploration that selects configurations where the model is uncertain about the performance.

Auto-sklearn generalizes this concept further so that at first it selects the machine learning algorithm and preprocessing steps. Afterwards the corresponding hyperparameters are tuned.

The end result is an ensemble of classifiers. This greedily generated ensemble selects iteratively the classifier from the set of all trained classifiers that improves the performance of the previously selected classifier the most.

2.3.4 Measuring Classifier Performance

The following gives an overview of common measurements we used to determine the performance of a classifier.

- Precision: It is the ratio of true positive instances to predicted positive instances.
- Recall: Recall is the ratio of predicted true positive instances to actually positive instances.
- f1-score: This score is the harmonic mean of precision and recall.
- Receiver operating characteristic curve: That is a graphical plot that illustrates the diagnostic ability of a binary classifier by varying its discrimination threshold.
- Learning curve: It is a graphical representation of how the model performance changes by training it on a steady increasing training set.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Accuracy: Accuracy is the ratio of the number of correctly predicted instances to the total number of tested instances.

Precision, Recall, and f1-score can be calculated either by macro-average or by micro-average. Macro-average computes the metric independently for each class and then take the average (hence treating all classes equally), whereas micro-average aggregates the contributions of all classes to compute the average metric.

2.3.5 K-fold Cross Validation

This method is investigated by Rodríguez et al. [48] and is used to obtain a better estimation of model prediction performance. This method divides a dataset D into k portions P_i for $1 \leq i \leq k$. Afterward, k models are trained with the same machine learning technique. The i^{th} model is trained on the data set $D \setminus P_i$ and tested on the portion P_i. The average of all test results is the end result of the k-fold cross-validation. This method tries to reduce overfitting since the whole data set is considered for testing and not a fixed portion. The disadvantage of this method is that it requires k times more run time than a traditional training/test set approach.

2.3.6 Hyper Parameter Tuning

Often machine learning methods have parameters that are not learned from a data set. They are called hyperparameters and in the past, they are often assigned manually by a machine learning expert. It is also possible to use computing power to determine good values for hyperparameters.

Two established hyperparameter tuning methods are grids search and random search as explained in Bergstra and Bengio [4]. These methods rate hyperparameter configurations by training a machine learning model with these parameters. Afterwards the performance of the trained models is checked according to the required needs. The remaining question is how the hyperparameter search space is explored.

Grid Search

If necessary this technique discretizes continuous hyperparameters into equidistant points. Afterward, all possible combinations of the discretized search space are explored.

Random Search

This technique uniformly picks samples from each hyperparameter space.

Bergstra and Bengio [4] argue that random search is more efficient than grid search if processing time is a limiting factor for the development of the machine learning model. Grid search has the issue that unimportant hyperparameters which do not have a significant effect on the model's performance adds a lot of hyperparameter configurations that lead to similar models.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.3.7 Feature Evaluation

Shapley Additive exPlanations (SHAP) from Lundberg and Lee [38] is a unified approach to explain the output of any machine learning model.

At a glance, SHAP takes a complex model, which has learned non-linear patterns in the data and breaks it down into lots of linear models that describe individual instances.

SHAP measures the importance of a feature based on Shapley values (see Shapley [51]). This concept originates in the field of cooperative game theory. In the binary classification setting a shapley value $\phi_i(p)$ for a certain feature i out of total feature set N, given the prediction p by the complex model is:

$$\phi_i(p) = \sum_{S \subseteq N/i} \frac{|S|! \cdot (|N| - |S| - 1)!}{|N|!} \cdot (p(S \cup i) - p(S)) \tag{2.2}$$

Equation (2.2) gathers the prediction of the model without feature i and also calculates the prediction of the model with feature i, then the difference between these two is obtained. However, the importance of a feature depends on the group of the remaining considered features. A Shapley value considers this by calculating a weighted sum of all possible sets S of feature groupings minus the feature i we are interested in.

The influence of a feature can be determined by calculating the absolute mean value of the Shapley values of all considered instances and comparing the result to the results of the other features.

2.4 Experiments

All experiments have been performed on a machine with an i5-8350U CPU and 16GB RAM. Scikit-learn from Pedregosa et al. [46] is used as a machine learning framework.

2.4.1 Evaluation Setup

Figure 2.1 illustrates and explains the used setup to obtain a good classifier.

2.4.2 Baseline

A classifier where the prediction is based on the distribution of the target classes in the training set is used as the baseline for other models. Since we use stratified training and test sets the baseline classifier predicts, with a probability of approximately 77%, the class *ok* independent of the given instance features values.

2.4.3 Experimental Results

The best-obtained classifier is a random forest that predicts based on the extended feature set. The following data preparation steps are applied in the given order:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

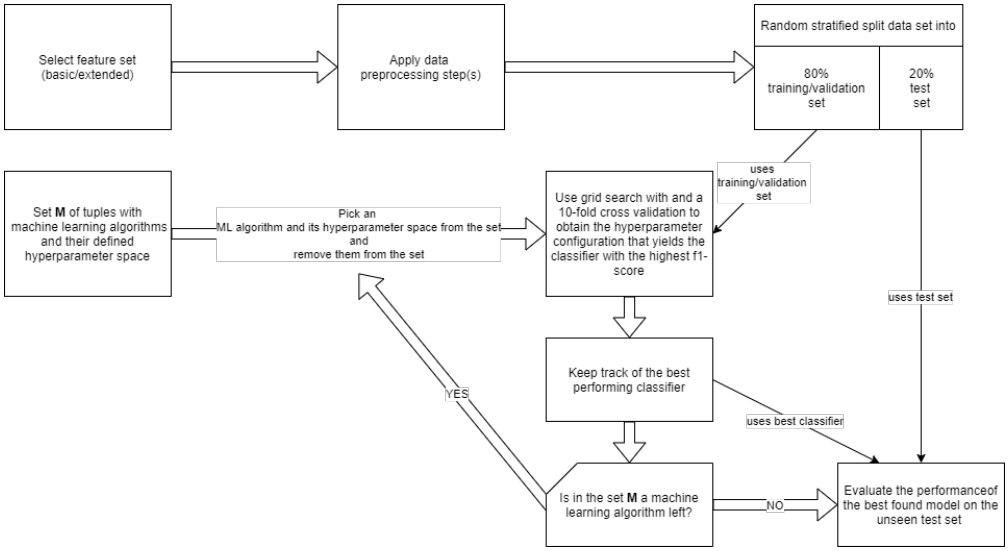


Figure 2.1: Experimental setup: The setup contains the following steps. The user selects either the basic feature set or the extended feature set for the generation of the machine learning data set. Further, the user decides which machine learning techniques plus the corresponding hyperparameter spaces are explored. Then the defined data preparation steps are applied to the machine learning data set. The resulting data set is split into 80% training set and 20% test set. All defined machine learning techniques are applied to the training set with a grid search for hyperparameters and 10-fold cross-validation for the evaluation of the f1-score performance. The machine learning model with the highest f1-score of all machine learning techniques is evaluated with different performance metrics.

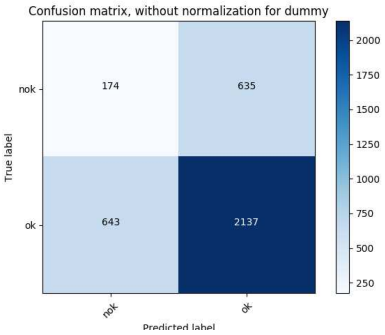
- The dimensionality of the one-hot encoded features related to the lacquers and the part types that are mounted on the skid is reduced with PCA where the number of components is set to 15.
- All feature values are scaled linearly between $[0, 1]$ according to Equation (2.1)

For the random forest classifier, the best-found hyperparameter configuration is:

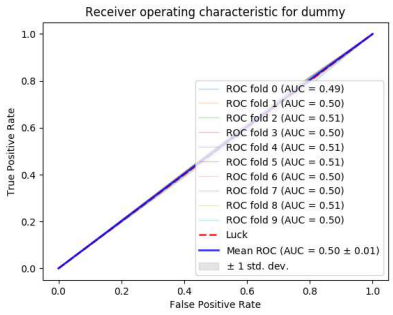
- Number of trees: 1000
- Percentage of considered features to select the best split: 20%
- The maximum depth of the trees: 70

All other hyperparameters for the random forest algorithm use the default values of the implementation of scikit-learn version 1.1.0.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN



(a) Confusion Matrix



(b) Learning Curve

Accuracy = 0.6439

	Precision	**Recall**	**F1-score**	**support**
nok	0.21	0.22	0.21	809
ok	0.77	0.77	0.77	2780
avg / total	0.65	0.64	0.64	3589

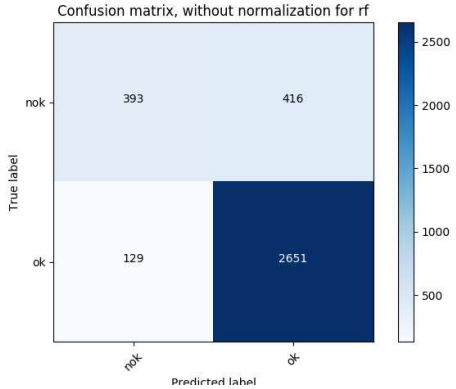
Figure 2.2: Measurements of the used baseline classifier

The performance metrics measured on the best found classifier are shown in Figure 2.3. Figure 2.2 contains the experimental result from the baseline classifier. The best found classifier is superior to the baseline classifier in all measured performance aspects.

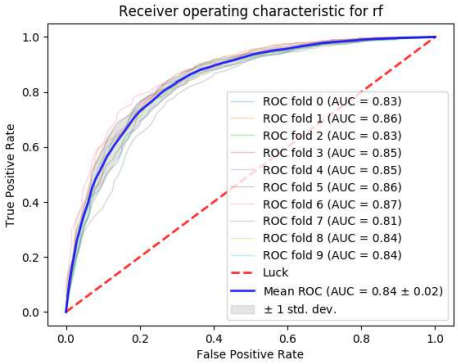
Figure 2.3b shows that the performance of the best found random forest model becomes better the more instances it gets to learn, but Figure 2.3a shows that a high portion of the *nok* instances from the test set are wrongly classified as *ok*. Therefore, we try to compensate this issue by turning the best found non-cost sensitive classifier into a cost-sensitive classifier with the concept as described in Section 2.3.3. The value of the cost matrix entry $C_{nok,ok}$ is fixed to 1. We investigated different values for $C_{ok,nok}$. The best outcome yields the assignment $C_{ok,nok} = 1.5$. The results of this cost-sensitive classifier are depicted in Figure 2.4. The cost sensitive classifier yields similar results compared to the performance of the best found non-cost sensitive classifier. In the confusion matrix Figure 2.4a it can be seen that the cost sensitive model predicts the label *nok* more often than the normal classifier in Figure 2.3a. The effect is that more true *nok* instances are detected correctly while the number of misclassified *ok* instances increases. If the value for $C_{no,nok}$ is further increased to 4 then the model predicts *nok* more likely. In this case, the accuracy is reduced drastically as it can be seen in Section 2.4.3. This assignment can be useful for example if it is very important to use a schedule that reduces the number of inferior lacquered parts.

The automated machine learning tool auto-sklearn is used to obtain a classifier. The performance of the classifier found by auto-sklearn can be seen in Figure 2.6. The best manually found model (Figure 2.3) and the best automatically found model (Figure 2.6) have very similar performance on the investigated metrics.

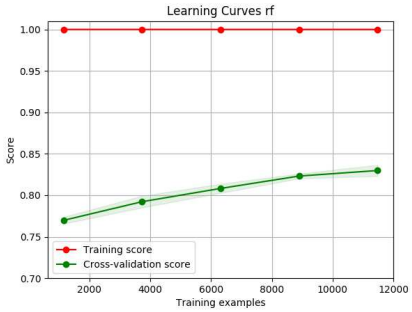
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN



(a) Confusion Matrix



(b) Receiver Operating Characteristic Curve

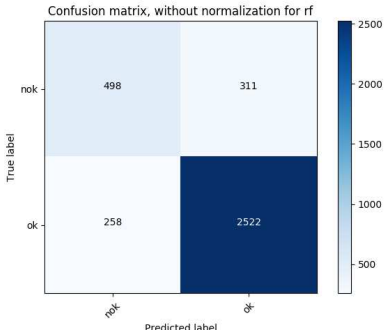


(c) Learning Curve

Accuracy = 0.8481

	Precision	**Recall**	**F1-score**	**support**
nok	0.75	0.49	0.59	809
ok	0.86	0.95	0.91	2780
avg / total	0.84	0.85	0.84	3589

Figure 2.3: Measurements of the best found machine learning pipeline



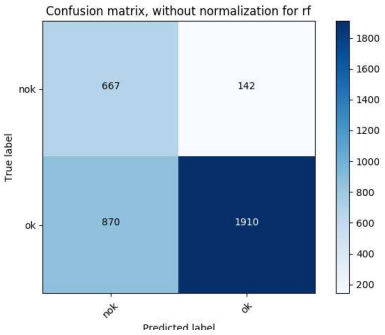
(a) Confusion Matrix

Accuracy = 0.8414

	Precision	**Recall**	**F1-score**	**support**
nok	0.66	0.62	0.64	809
ok	0.89	0.91	0.90	2780
avg / total	0.84	0.84	0.84	3589

Figure 2.4: Measurements of the cost sensitive classifier where $C_{ok,nok} = 1.5$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

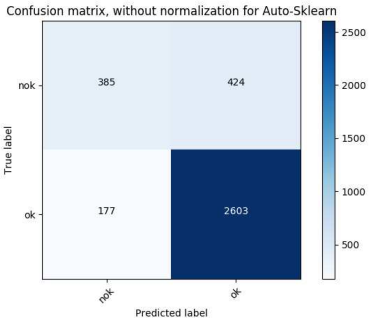


Accuracy = 0.7180

	Precision	**Recall**	**F1-score**	**support**
nok	0.43	0.82	0.57	809
ok	0.93	0.69	0.79	2780
avg / total	0.82	0.72	0.74	3589

(a) Confusion Matrix

Figure 2.5: Measurements of the cost sensitive classifier where $C_{ok,nok} = 4$



Accuracy = 0.8484

	Precision	**Recall**	**F1-score**	**support**
nok	0.71	0.55	0.62	809
ok	0.88	0.93	0.91	2780
avg / total	0.84	0.85	0.84	3589

(a) Confusion Matrix

Figure 2.6: Measurements of a classifier created by Auto-sklearn

2.4.4 Feature Evaluation

The tool SHAP as it is introduced in Section 2.3.7 is used to gain some insights about the influence of individual features on the prediction. The results of the calculated absolute mean SHAP values are evaluated in Figure 2.7.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

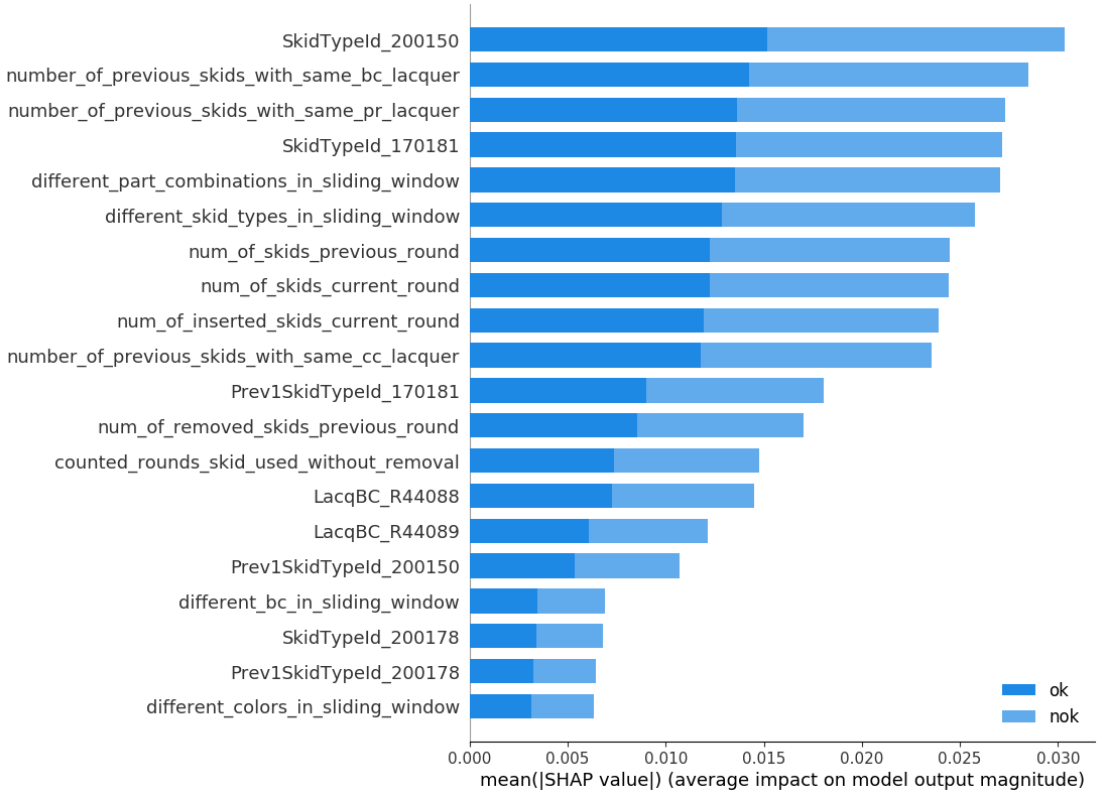


Figure 2.7: Average feature impact on model output based on the mean absolute SHAP values: For this experiment, the best performing random forest classifier is used. According to SHAP, the most important feature is about if the current skid is of type 200150. There is no obvious explanation for this since we do not have information about this skid type. This will be addressed in future work. The second and the third most important feature are: How many previous skids are lacquered with the required base- and primer coating of the current skid. This is reasonable since as identified beforehand frequent lacquer changes are a cause of inferior quality. The fourth most important feature is the check if the current skid type is 170181.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Automated Configuration of Machine Scheduling Dispatching Rules by Machine Learning

In this chapter we investigate automated configuration of machine scheduling dispatching rules by machine learning. Section 3.1 defines the investigated machine scheduling problem and provides a general background on dispatching rules. Later in Section 3.2, we give an overview of current state-of-the-art approaches to tune dispatching rules for parallel machine scheduling. Section 3.3 introduces the novel instance-specific features and describes the proposed machine learning approach in detail. Finally, Section 3.4 summarizes the conducted experimental results.

3.1 Dispatching Rules for Parallel Machine Scheduling

In this section we first give a formal specification of the Parallel Machine Scheduling problem (PMSP) in Section 3.1.1 and afterwards provide some background on dispatching rules in Section 3.1.2.

3.1.1 Parallel Machine Scheduling

The goal of a PMSPs is to find an optimized assignment of a given set of jobs to a set of machines machines so that a number of cost objectives is minimized. In the following we describe a traditional problem formulation of the PMSP as it has been thoroughly studied in the literature (e.g. Mokotoff and Ethel [40]).

Instance parameters

Several parameters define instances of the PMSP:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

$\mathcal{H}$ set of periods in the scheduling horizon

P set of Machines

J set of Jobs

M set of Materials

$b_j \in M, \forall j \in J$ job j processes material b_j

$E_j \subseteq P,\ E_j \neq \emptyset, \forall j \in J$ non empty set of eligible Machines for job j

$t_j \in \mathcal{H}, \forall j \in J$ due date of job j

$d_{j,p} \in \mathbb{N}, \forall j \in J\ \forall p \in P$ process duration of job j on machine p

$s_{m',m,p} \in \mathbb{N}$ setup time for a job with material m after a job with material m' is executed on p

Decision variables

Two sets of decision variables define a solution to the PMSP:

$st_j \in \mathcal{H}$ start time for job j

$sp_j \in P$ job j is scheduled on machine sp_j

Hard Constraints Two hard constraints need to be fulfilled in a solution to the PMSP:

- If a job j is scheduled on the machine sp_j and the job j' is executed before j on sp_j then sp_j is occupied for the time interval $[st_j, st_j + s_{b_{j'},b_j,p} + d_{j,p}]$. During this time no other job can be executed on the machine sp_j.
- All jobs j are scheduled on a machine $p \in E_j$.

Solution Objectives The following solution objectives describe the cost function of the PMSP ($previousJob(e,p)$ denotes the latest job that is scheduled before time e on machine p):

- Minimize the number of due date violations

$$\sum_{j \in J} \begin{cases} 1, & \text{for } st_j + d_{j,sp_j} + s_{b_{previousJob(st_j,sp_j)},b_j,sp_j} > t_j \\ 0, & \text{otherwise} \end{cases}$$

- Minimize the total tardiness of jobs which are completed after the due date

$$\sum_{j \in J} max(st_j + d_{j,sp_j} + s_{b_{previousJob(st_j,sp_j)},b_j,sp_j} - t_j, 0)$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Minimize the make span

$$max(\{st_j + d_{j,sp_j} + s_{b_{previousJob(st_j,sp_j)},b_j,sp_j} : j \in J\})$$

- Minimize the total setup time for all jobs

$$\sum_{j \in J} s_{b_{previousJob(st_j,sp_j)},b_j,sp_j}$$

- Minimize the number of setup processes

$$\sum_{j \in J} \begin{cases} 1, & \text{for } s_{b_{previousJob(st_j,sp_j)},b_j,sp_j} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

3.1.2 Configurable Dispatching Rules for Machine Parallel Machine Scheduling

Dispatching rules for the PMSP produce solutions for a given problem instance by greedily assigning the jobs to machines one by one. In configurable dispatching rules, a given set of n parameters $W = \{w_1, \ldots, w_n\}$ determines in which order the job assignments are conducted.

During its execution, a dispatching rule uses a priority function to calculate priority values for each of the pending job assignments. These priority values depend on the partial solution's state, the job attributes, as well as the dispatching rule parameters. The dispatching rule then chooses the job assignment with the highest priority value in each iteration. Often, the priority function simply calculates a weighted sum of a list of attributes using the weights $w_i, i \in \{1, \ldots n\}$ to rank the pending job assignments.

Figure 3.1 depicts an example procedure of a dispatching rule and visualizes some dispatching iterations. An overview on different ways of computing of priority values that have been studied in the literature can be found for example in Panwalkar and Iskander [44], Haupt [26], and Blackstone et al. [5].

Note that dispatching rules usually guarantee to create feasible solutions that do not cause any hard constraint violations. However, as jobs are greedily assigned to machines, dispatching rules often cannot be used to obtain global optimal solutions.

Figure 3.2 visualizes an example where a dispatching rule aims to minimize total make span with the use of a weighted sum priority function that incorporates evaluation of total setup times. This exemplifies how greedy dispatching cannot always be configured to find global optimal solutions.

Note that a selection of good parameters is crucial for the performance of any dispatching rule. In practice this is usually done by a human expert that aims to find efficient parameters by manually performing a series of simulation runs with a given dispatching rule. In this work we aim to automatize the process of finding the best parameter configuration for a given instance and dispatching rule, even if it is not guaranteed that we can find global optimal solutions due to the greedy nature of these rules.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

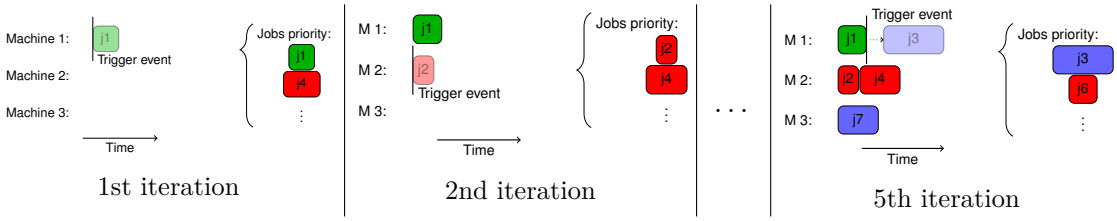
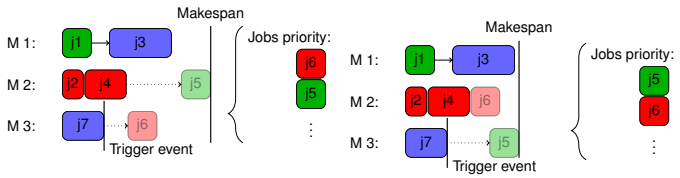


Figure 3.1: Dispatching Rule Planning Procedure
In this figure, the schedules for each machine are depicted row-wise and jobs are visualized as colored boxes. The horizontal length of each job's box corresponds to the job duration, while the color of each job encodes the processed material. Furthermore, the length of the horizontal arrows between two jobs indicates the required setup time.
At each iteration, a dispatching rule determines the earliest possible starting time e on a machine p for the next to be scheduled job. We call the tuple (e, p) the trigger event of an iteration. The dispatching rule then calculates priority values for each job and schedules the job j with the highest priority function. In other words, job j will be scheduled to start on machine p at time e. The procedure continues until all jobs are scheduled.



(a) Unimproved dispatching rule (b) Improved dispatching rule

Figure 3.2: Local optima problem:
This example is based on the partial schedule from Figure 3.1. In Figure 3.2a the example dispatching rule tries to minimize total make span by using a setup time weight parameter in its priority function. In the current iteration the dispatching rule selects a pending job to be scheduled on machine 3, as it is available at the earliest time in the current schedule. Therefore, the job with id 6 is scheduled at machine 3 as it seemingly leads to a smaller setup time. However, to find the globally minimal make span it would be preferable to schedule the job with id 5 first as it is shown in Figure 3.2b.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.2 Literature Review

In the literature various methods for finding good dispatching rules have been proposed. An overview about finding good rules is given by Branke et al. [7]. The automated parameter configuration part of the thesis focuses on using hyper-heuristics to find a good priority function for dispatching rules that work well on all instances of a particular instance set. Two representations are presented:

- Fixed-Length Parametric Representations: In this approach the basic format of the priority function is predefined. Parameters such as scalar factors represent the dimensions of the search space and the goal is to find good parameter *values*. Our parameter configuration search space for dispatching rules belongs to this type.
- Variable-Length Grammar-Based Representations: This is an alternative way of defining the search space of priority functions that is based on a grammar that specifies how the individual components can be assembled. The components are attributes of a job and mathematical functions like $+$, $\times$, min, max.

Branke et al. [8] compare these two representations with and without normalized attributes. They conclude that normalization improves the performance of hyper-heuristics for both representations. Variable-Length Grammar-Based Representations yields a better performance but they need more iterations compared to the Fixed-Length Parametric Representations to find a good configuration.

Heger et al. [27] improve dispatching rule-based scheduling by switching between dispatching rules dynamically depending on the current system conditions. If, for example, a machine is highly utilized, a rule that avoids longest setup times is preferred. Otherwise, a rule that selects the most urgent jobs is applied (this should lead to a lower mean tardiness). The remaining issue is how to select the right rule at event time e (given a partial schedule, the event time e is the earliest possible time when any of the machines becomes available in the schedule). To achieve this, in Heger et al. [27] a Gaussian process regression model was applied.

Mouelhi-Chibani and Pierreval [43] use a similar technique. In this paper a neural network is used to select a rule dynamically. The neural network is trained through a simulation-optimization approach that uses a physical model of the workshop and generated random jobs. The considered problem in the paper is a simplified flow-shop with two work-centers WC1 and WC2, where each work-center has two machines that perform the same task. Each job has to be processed first on one machine at WC1 and then on one machine at WC2. Each work-center has a waiting queue in case both machines are busy. A job remains in the waiting queue until a machine becomes idle.

El-Bouri et al. [17] also use neural networks to create dispatching rules. The attributes of jobs are used as features for the neural network. The neural network decides how the attributes are combined to form a prioritization function.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Burke et al. [12] give a general overview about hyper-heuristics which contains also a section about scheduling problems. The presented methodologies deals with selecting good dispatching rules out of a set of defined rules.

Parkes et al. [45] propose a method to predict if the given parameter configuration is good or not for a heuristic that is used for the online bin packing problem. This is done by a Multi-layer Perceptron model. The outcome of the model is a binary classification with two classes: *good parameter configuration* or *bad parameter configuration*. The input of this model is based on the parameter values.

Our aim in the automated parameter configuration part of the thesis is to automatically find best parameters for dispatching rules based on the features of a particular problem instance. To this aim we propose an extensive set of features to characterize the problem instances and train a regression model that is able to predict a parameter configuration for dispatching rule weights based on instance features.

3.3 Automated Parameter Configuration for Dispatching Rules

In this section we propose an automated configuration of dispatching rules with supervised machine learning. The workflow of our proposed approach is outlined as follows:

Section 3.3.1 explains how we generate parallel instances with the use of a random instance generator that later serve as training and validation data sets for machine learning. Afterwards, in Section 3.3.2 we propose features that characterize a parallel machine scheduling instance. Section 3.3.3 then describes how the ground truth target values for the data set are obtained. Since for the parameter configuration of dispatching rules we usually need to select multiple continuous values, we need multi-target regression machine learning models which are described in Section 3.3.4.

3.3.1 Generated Parallel Machine Scheduling Instance Set

To generate a large set of instances we use a generator based on random generators from the literature Vallada and Ruiz [54] and Moser et al. [42].

The instance generator given in Algorithm 1 takes as input the desired number of machines, jobs and materials and returns a randomly generated instance of the parallel machine scheduling problem. Additionally, there is another boolean input parameter called *longerDurations* which determines the ranges of the uniform distributions that are used to sample job durations and setup times (see lines 2-7 in Algorithm 1). If $longerDurations = False$ (or $longerDurations = True$), then the following distributions are used:

- job durations: A uniform distribution with a value range from 1 to 99 (or 1 to 124) is used to draw samples for the job durations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 1 Instance generator

procedure GENERATE INSTANCE(number of machines $|P|$, number of jobs $|J|$, number of materials $|M|$, boolean flag *longerDurations*)
  if longerDurations **then**
    $upBoundSetupTime = 99$
    $upBoundDuration = 124$
  else
    $upBoundSetupTime = 124$
    $upBoundDuration = 99$
  $P = \{1, \ldots, |P|\}$
  $J = \{1, \ldots, |J|\}$
  $M = \{1, \ldots, |M|\}$
  fac = vector of size $|P|$

    where each element is drawn from the continuous uniform distribution $U(0.5, 1.5)$
  $setupTime$ = 3 dimensional tensor $T_{i,j,k}$ with size $|M| \times |M| \times |P|$

    where each element value is drawn from the discrete uniform distribution $U\{1, upBoundSetupTime\}$

    if $i \neq j$ otherwise the value is set to 0.
  b = vector of size $|J|$
  d = matrix of size $|J| \times |P|$
  E = array of size $|J|$ where the elements are subsets of P
  $previousMaterial$ = zero vector of size $|P|$
  $busyUntil$ = zero vector of size $|P|$
  for each $j \in J$ **do**
    $b_j = \begin{cases} j, & \text{for } j \leq |M| \\ \text{sample of } U\{1, |M|\}, & \text{otherwise} \end{cases}$
    if $j \leq |P|$ **then**
      $lowBoundE = 0$
      $upBoundE = |P| - 1$
      $E_j = \{j\}$
    else
      $lowBoundE = 1$
      $upBoundE = |P|$
      $E_j = \{\}$
    $cardinality$ = sample from $U\{lowBoundE, upBoundE\}$
    $E_j = E_j \cup$ subset of P with random selected elements and cardinality $cardinality$
    $duration$ = sample from $U\{1, upBoundDuration\}$
    $d_j = duration \cdot fac$
    $machine = \underset{p \in E_j}{arg\ min}(busyUntil_p + setupTime_{b_j, previousMaterial_p, p} + duration_{j,p})$
    $d_j = busyUntil_p + setupTime_{b_j, previousMaterial_j, p} + duration_{j,p}$
    $busyUntil_p = d_j$
    $previousMaterial_p = b_p$
  return parallel machine problem instance with $P, J, M, b, E, t, d, setupTime$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- setup times: A uniform distribution with a value range from 1 to 124 (or 1 to 99) is used to draw samples for the setup times. The setup times between jobs with the same material are fixed to 0.

The vector fac (line 11) defines for each machine a random factor between 0.5 and 1.5.

Setup times between jobs are set in line 12 as a three dimensional tensor. The first dimension represents the previous job, the second one represents the current job and the last dimension the machine.

Line 13, 14, 15 declare instance variables. The first one is the mapping from jobs to materials. Followed by the due dates d and eligible machines E for all jobs.

The generated job to material mapping guarantees that every material is used at least for one job (line 19). All jobs with a smaller or equal identification number than the total number of materials are directly mapped to the material that has an id that is equal to the job id. For example, if we have 5 materials then for the first five jobs the materials with ids 1-5 will be assigned. Any remaining jobs get a random material number between 1 and 5.

A similar technique is used in lines 20-27 to guarantee that every machine is eligible for at least one job.

For each job that has an id that is smaller or equal to the total number of machines, the machine that has the same id will be included in the set of eligible machines. For all jobs the *cardinality* of the eligible machine set it determined by a random number. For jobs with an id that is smaller than the number of machines, the number of eligible machines will be set to a random number between 0 and $|P| - 1$. Here 0 is used as a lower bound and $|P| - 1$ is used as an upper bound since these jobs have in any case at least one eligible machine. For the remaining jobs the random number ranges from 1 to P. *cardinality* many machines are then selected from the possible machines P. The obtained subset is the set of eligible machines for a job.

Line 30 samples for each job a *duration* value from a uniform distribution.

In line 31 the previously set factors fac are multiplied with the *duration* value to define the per machine durations for a single job.

Furthermore, to obtain tight but still feasible due dates it is necessary to construct a feasible schedule. Therefore, the generator constructs a solution with no due date violations. The helper variables defined in 16-17 are required to construct a schedule, where the variable $previousMaterial$ contains for each machine the material of the last scheduled job and the variable $busyUntil$ contains for each machine the earliest starting time of any pending job assignment.

In line 32 the next job is scheduled on the machine that allows the earliest completion time. Line 33 then assigns this completion time to be the due date of this job. This procedure continues until all jobs have an assigned due date. The lines 34-35 take care of the machine states. Line 36 returns the generated instance.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.3.2 Features for a Parallel Machine Scheduling Instance

In this section we propose a set of features that characterize parallel machine learning instance, and can be used for an automated dispatching rule parameter configuration with machine learning.

We use the following terms in the definition of the features:

$$\begin{aligned} avgJobToJobSetupTime_{j',j} &= \text{avg}(\{s_{j',j,p} : p \in E_j \cap E_{j'}\}) \\ avgJobSetupTime_j &= \text{avg}(\{avgJobToJobSetupTime_{j',j} : j' \in J, E_j \cap E_{j'} \neq \emptyset\}) \\ avgSetupTime &= \text{avg}(\{avgJobSetupTime_j : j \in J\}) \end{aligned}$$

$$\begin{aligned} medJobDuration_j &= \text{median}(\{d_{j,p} : p \in E_j\}) \\ medDuration &= \text{median}(\{medJobDuration_j : j \in J\}) \end{aligned}$$

The list of features that we propose for the automated parameter configuration part of the thesis are specified as follows:

- Ratio of number of jobs to number of machines:

$$\frac{|J|}{|P|}$$

- Ratio of number of material to number of machines:

$$\frac{|M|}{|P|}$$

- Relative standard deviation of the job durations: The medians of the durations for each job over all possible machines is taken separately and the relative standard deviation of all these values is used as a feature. The relative standard deviation (RSD) is the ratio of the standard deviation to the mean.

$$\text{RSD}(\{\text{median}_{p \in E_j}(d_{j,p}) : j \in J\})$$

- Uniform distribution of due dates: The Kolmogorov-Smirnov test for goodness of fit is used. It tests the distribution of the due dates against a uniform distribution $U(earliestduedate, latestduedate)$. Under the null hypothesis, the two distributions are identical. The resulting p-value of this test is used as a feature.

- Skewness of due dates: The skewness of the due date distribution varies depending on whether or not there are more jobs with an early due date compared to jobs with a late due date or vice versa. We therefore use the Fisher-Pearson coefficient of skewness of the due date distribution as feature.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Ratio of minimum number of eligible machines over all jobs to number of machines:

$$\frac{\min_{j \in J} E_j}{|P|}$$

- Ratio of average number of eligible machines over all jobs to number of machines:

$$\frac{\text{avg}_{j \in J} E_j}{|P|}$$

- Relative standard deviation number of eligible machines over all jobs

$$\text{RSD}(\{|E_j| : j \in J\})$$

- Ratio of average setup time to median job duration

$$\frac{avgSetupTime}{medDuration}$$

- Relative standard deviation of the average setup time for each job:

$$\text{RSD}(\{avgJobSetupTime_j : j \in J\})$$

- Ratio of estimated make span to last due date. The make span can be estimated by simply summing up average job setup time and median job duration for each job.

$$\frac{\frac{\sum_{j \in J}(avgJobSetupTime_j + medJobDuration_j)}{|P|}}{\max_{j \in J}(t_j)}$$

- Ratio of a lower bound to an upper bound on the required machine time
  The lower bounds for job setup times and durations are defined as follows:

$$\begin{aligned} minJobToJobSetupTime_{j',j} &= \min(\{s_{j',j,p} : p \in E_j \cap E_{j'}\}) \\ minJobSetupTime_j &= \min(\{minJobToJobSetupTime_{j',j} : j' \in J, E_j \cap E_{j'} \neq \emptyset\}) \\ minSetupTime &= \min(\{minJobSetupTime_j : j \in J\}) \end{aligned}$$

$$\begin{aligned} minJobDuration_j &= \min(\{d_{j,p} : p \in E_j\}) \\ minDuration &= \min(\{minJobDuration_j : j \in J\}) \end{aligned}$$

  The upper bounds are defined analogously to the lower bounds as $maxJobSetupTime_j$ and $maxJobDuration_j$.

$$feature = \frac{\sum_{j \in J}(minJobSetupTime_j + minJobDuration_j)}{\sum_{j \in J}(maxJobSetupTime_j + maxJobDuration_j)}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Ratio of average number of all pairwise eligible machine set intersections of all job pairs to the total number of machines.

$$\frac{\text{avg}(\{|E_{j_1} \cap E_{j_2}| : j_1, j_2 \in J, j_1 < j_2\})}{|P|}$$

- Relative standard deviation of all eligible machine intersections of all job pairs:

$$\text{RSD}(\{|E_{j_1} \cap E_{j_2}| : j_1, j_2 \in J, j_1 < j_2\})$$

Note that all proposed features are defined relative to scale attributes so that the the features are independent from the instance size. Therefore, structural instance information that has been learnt from small instance can be helpful to make predictions also on larger instances and vice versa.

3.3.3 Exhaustive Grid Search

Previously, in Section 3.3.1 and Section 3.3.2 we have described how parallel machine scheduling instances can be randomly generated and how instance specific features can be extracted. However, before we can use this data to train machine learning models for the automated parameter configuration of dispatching rules we further need to determine target values for each instance. In the following we describe a grid search approach that can be used to find an optimized parameter configuration for a given dispatching rule and instance. Furthermore, we show how this grid search can also be used to achieve normalized objectives within a multi-objective setting that is common for parallel machine learning problems later in Section 3.3.3.

We rate a parameter configuration by applying a multi-objective function to the solution which is returned by the dispatching rule that is configured with the given parameters. Therefore, we use an exhaustive grid search that enumerates the outcomes of a given dispatching rule with all possible parameter configurations for a single instance and then select a configuration that leads to the overall lowest objective value. The selected parameter configuration then is used as the target value for the corresponding instance.

To deal with non finite parameter domains (such as continuous value domains or unrestricted ranges) we use a discretization procedure to convert such domains into finite domains that can be enumerated. Later in Section 3.4 we provide a detailed discretization example.

In the following sections we refer to the set of all possible parameter configurations as $\mathcal{C}$ and denote the costs of the solution returned by the corresponding dispatching rule with a parameter configuration $c \in C$ as $s(c)$.

Normalizing Multi-Objective Cost Functions

Parallel machine scheduling problems usually aim to minimize multiple objectives like for example make span, total tardiness and the total setup times. Since some of these

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

objectives are potentially in conflict with each other, often there does not exist a solution that can lower all objectives to the lowest possible value at the same time. A solution in the multi-objective optimization context is Pareto optimal if there exists no other feasible solution that improves the value of at least one objective function without declining any other objective. The set of Pareto optimal solutions forms a Pareto optimal set.

The following demonstrates how the five objectives $(o_1, o_2, o_3, o_4, o_5)$, that have been previously defined in Section 3.1 are normalized for an instance *inst* of the parallel machine scheduling problem. We define the individual minimal objective function value for each $o_i, i \in \{1, \ldots, 5\}$ as follows:

$$min_i = \min_{c \in \mathcal{C}} o_i(s(c))$$

min_i for $i \in \{1, \ldots, 5\}$ forms the vector min, which is called the utopia point by Mausser [39]. Usually, no feasible solution that has a cost vector equal to the utopia point exists because of the conflicting nature of the individual objectives. However, it can be used to provide lower bounds on the Pareto optimal set.

Furthermore, we denote as $s_i, \forall i \in \{1, \ldots, 5\}$ a single solution where $o_i(s_i) \leq o_i(s(c)), \forall c \in C$ holds. Note that there are potentially multiple solutions that could be used as s_i, however we select only a single solution for our purposes (in our implementation we simply use the first encountered solution that has minimal costs regarding o_i).

Using solutions $s_1, \ldots, s_5$ we further define corresponding maximal objective values $max_i, \forall i \in \{1, \ldots, 5\}$:

$$max_i = \max_{j \in \{1,\ldots,5\}, j \neq i} o_i(s_j)$$

max_i for $i \in \{1, \ldots, 5\}$ also forms a vector denoted by max. Mausser [39] calls the parameter configuration which yields max_i for $i \in \{1, \ldots, 5\}$ a nadir point.

Mausser [39] uses the utopia and nadir points to normalize the individual objectives in a multi objective setting as follows:

$$o_i^{norm}(c) = \frac{o_i(s(c)) - min_i}{max_i - min_i}$$

However, for our purposes we cannot use this procedure as we cannot guarantee that $max_i - min_i = 0$ which would lead to an undefined result because of a division by 0. To handle our extended normalization requirements, we therefore adapt the normalization procedure from Mausser [39] as follows:

$$o_i^{norm}(c) = \begin{cases} 0, & \text{for } o_i(s(c)) - min_i = 0 \\ \frac{o_i(s(c)) - min_i}{\frac{(max_i - min_i) + (o_i(s(c)) - min_i)}{2}}, & \text{otherwise} \end{cases}$$

Instead of using a division by $max_i - min_i$, we divide by the arithmetic mean of the difference between the corresponding maximal and minimal values and the difference

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

from the actual cost value to avoid a division by zero if $max_i - min_i = 0$. Additionally, we have to handle a rare corner case where $max_i - min_i = 0$ and $o_i(s(c)) - min_i = 0$. In this case we simply set the normalized objective to 0, as the objective lies exactly on the Pareto front.

The sharp eyed reader might have noticed that we mentioned earlier at the beginning of Section 3.3.3 that we use a grid search approach to find the Pareto values. This however does not cover the complete configuration space C due to the discretization that became necessary to practically implement the grid search. Therefore, it is possible that objective values better than the previously computed utopia points can be achieved with a configuration $c \in C$. This however poses no problem for the normalization routine as o_i^{norm} would simply be smaller than 0 in such a case.

Finally, we use the individually normalized objective values to define an overall normalized objective function o^{norm} using a weighted sum of the individual normalized objectives, where $w_i, \forall i \in \{1, \ldots, 5\}$ are user defined weights:

$$o^{norm}(c) = \sum_{i \in \{1,\ldots,5\}} w_i \cdot o_i^{norm}(c) \tag{3.1}$$

In the following sections we use o^{norm} to rate dispatching rule parameters for a a given parallel machine scheduling instance based on a given set of weighted objectives.

Finding Optimized Target Values

For each instance *inst* of the parallel machine scheduling problem we define the set of optimal parameter configurations c_{min} as:

$$c_{min} = arg \min_{c \in \mathcal{C}}(o^{norm}(c))$$

In practice we can find the set of c_{min} by executing the previously defined grid search on the discretized configuration space in two iterations. The first iteration is necessary to obtain the function o^{norm} as described in Section 3.3.3. Afterwards, a second grid search iteration can be used to determine c_{min} for the discretized configuration space.

This procedure could in principle be used to determine high quality parameter configurations. However, in practice it is often too computationally expensive to conduct two grid search iterations whenever parameter configurations for a new instance need to be found. Therefore, in the following section we describe how a machine learning approach can be used to predict parameter configurations for a given instance. To determine labels for the training set, we simply assign any configuration c that is contained in c_{min} as the ground truth target vector for each instance (in our implementation we use the first encountered configuration in c_{min} as ground truth value).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.3.4 Machine Learning

Since a dispatching rule can have multiple continuous parameters we use a multi-target regression model (Borchani et al. [6]) to predict a good parameter configuration for a given instance based on its characteristics. For each created instance of Section 3.3.1 the defined features from Section 3.3.2 then form the input of the multi-target regression model.

Using machine learning to predict an optimized configuration for an unseen instance has the advantage that no dispatching rule has to be executed and therefore can be expected to be significantly faster than the previously described grid search approach in practice.

Multi-Target Regression Models

Borchani et al. [6] give a survey about state-of-the-art multi-target regression models. A simple way to obtain a multi-target regression model is to train a separate model for each target. However, a disadvantage of this approach is that potential dependencies between the target variables are not considered.

Louppe [37] describes a common technique to address also dependencies between the targets is to extend a random forest regressor by using leafs that store multiple output values and computing the splitting criteria based on the average reduction from all reachable outputs. Another technique that has been suggested in the literature is a k-nearest neighbors regressor that predicts multiple values by considering the average target values of the k-nearest neighbors. This technique is used in the machine learning framework sklearn from Pedregosa et al. [46]. Furthermore, neural networks can be extended to a multi-target regressor by using more than one node at the output layer. For example this technique is described by Specht [52] and used by An et al. [1].

3.4 Experimental Evaluation

To evaluate our method we focus on a real-life scenario that arises in industrial applications. This scenario is based on a greedy dispatching rule that is described in the following.

3.4.1 A Greedy Dispatching Rule

The used greedy dispatching rule uses a linear weighted sum of job attributes to prioritize the jobs. We call this specific dispatching rule in the subsequent sections a weighted rule. It works in the following way:

Assume we want to obtain the prioritization value at event time e for the unscheduled job j on machine p. Let j' be the job that was previously scheduled on machine p. Then the following attributes are used for the dispatching rule to prioritize the jobs.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

due date: Jobs with an earlier due date have higher priority. This attribute is defined as follows:

$$c_{due\_date} = \frac{e}{t_j}$$

critical ratio: The ratio between job duration and time left until its due date. Jobs with a higher ratio have a higher priority.

The formal definition of this attribute is given below.

$$c_{critical\_ratio} = \frac{d_{j,p}}{t_j - e}$$

setup time: This criterion rates the quality of the setup time that occurs between jobs j' and j on machine p. To measure the quality of this setup time, we compare it with the possible minimum and maximum setup times for job j which are based on the set of the remaining unscheduled jobs S at event time e. These minimum and maximum setup times min_j^S and max_j^S are specified as follows:

$$min_j^S = \min_{j' \in S, p \in P} s_{b_{j'}, b_j, p}$$
$$max_j^S = \max_{j' \in S, p \in P} s_{b_{j'}, b_j, p}$$

The formula for the setup time attribute is given below:

$$c_{setup\_time} = 1 - \frac{s_{b_{j'}, b_j, p} - min_j^S}{max_j^S - min_j^S}$$

The shorter the setup time of a job j, the higher the attribute value and therefore also the prioritization of this job.

Note that the min_j^S and max_j^S values are adapted at each iteration of the weighted rule and therefore $c_{setup\_time} = 1$ always represents for all jobs their current shortest reachable setup time while $c_{setup\_time} = 0$ represents the current longest reachable setup time. This ensures a fair comparison between the setup times of different jobs.

The priority of a job is calculated as follows: Each attribute value is multiplied with a weight factor and then summed to obtain the overall priority value for a job.

$$\begin{aligned} priority = {} & w_{due\_date} * c_{due\_date} + \\ & w_{critical\_ratio} * c_{critical\_ratio} + \\ & w_{setup\_time} * c_{setup\_time} \end{aligned} \tag{3.2}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The weight factors $w_{due\_date}, w_{critical\_ratio}, w_{setup\_time}$ are parameters of the weighted rule. In our case this is a triple of weights that represent a parameter configuration $c \in \mathcal{C}$. For example one possible configuration could be $(w_{due\_date}, w_{critical\_ratio}, w_{setup\_time}) = (6, 1, 3)$.

When an exhaustive grid search is applied to tune this parameters we enforce the sum of the weights to be 10 $w_{due\_date}, w_{critical\_ratio}, w_{setup\_time}$ in Equation (3.2) is equal to 10. With this constraint, the whole search space is still explored since in our case the weights define a ratio between attributes. So every possible weight parameter configuration can be linear scaled such that the sum of the weights is equal to 10 while the ratio between the weights is preserved.

A grid search explores an approximation of the search space. The approximation is done by discretization:

$$w_{due\_date}, w_{critical\_ratio}, w_{setup\_time} \in \mathbb{N}$$
$$\text{with the constraint}$$
$$w_{due\_date} + w_{critical\_ratio} + w_{setup\_time} = 10$$

The number of possible parameter combinations can be calculated combinatorially and is equal to the binomial coefficient of $\binom{12}{10} = 66$.

3.4.2 Prioritization Values of the Multi-Objective Function

In practice often the most important objective is to minimize due date violations. Therefore, for experimental purposes we set in Equation (3.1) for the objective *number of due date violations* the weight to 1.5. The weight for all other objectives is set to 1.0.

3.4.3 Evaluated Multi-Target Regression Algorithms

We evaluated different multi-target regression approaches. These include random forest, multi-layer perceptron and k-NN. For all algorithms we tuned various hyperparameters.

For the random forest approach the following hyperparameters were tuned by random search in 50 search iterations:

- Number of trees: $\{200, 400, 600, \ldots, 2000\}$
- Number of features to consider at every split: $\{number\ of\ features, \sqrt{number\ of\ features}\}$
- Maximum depth of the trees: $\{no\ restriction, 10, 20, 30, \ldots, 110\}$
- Minimum number of samples required to split a node: $\{2, 5, 10\}$
- Minimum number of samples required at each leaf node: $\{1, 2, 4\}$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Use bootstrapping to select samples for training: $\{true, false\}$

The best found hyperparameter values for random forest are: number of trees = 2000, number of features to consider at every split = $\sqrt{number\ of\ features}$, maximum depth of the trees = 10, minimum number of samples required to split a node = 5, minimum number of samples required at each leaf node = 2 and use bootstrapping to select samples for training = *true*.

For the the k-NN algorithm we also tuned the hyperparameter k with random search (50 iterations). The search space for k is set to $\{5, 25, 45, 65, \ldots 2000\}$. The best results were obtained with $k = 45$

An useful property of the random forest and k-NN approaches is that the sum of the predicted output is always 10 in our experiments, as in both cases the average of the dispatching rule weights of a subset of instances is used for prediction. Each training target weight configuration sums up to 10, therefore this property holds for all averaged weights which makes it easy to compare with other approaches.

A multi-layer perceptron model was built with Keras. The architecture of the multi-layer perceptron model was fixed based on several experiments with different configurations. In the final experiment two dense hidden layers with 10 nodes/layer were used. The output layer contains three nodes and uses a softmax function that normalizes all outputs such that the sum of all predicted values is equal to 1. To compare to other techniques the predicted values are multiplied by 10. Therefore, in the training phase all the target variables of the original data set were divided by 10.

3.4.4 Other Evaluated Approaches

To evaluate the performance of our approach, we additionally examined the performance of three alternative methods.

We first applied the exhaustive grid search approach. This approach is implemented as described in Section 3.4.1. Further, we compared to a baseline approach that sets all weights to 3.33. In this case all attributes get equal importance when the dispatching rule prioritizes the jobs.

Automated Algorithm Configuration We also investigated the performance of an automated algorithm configurator. In our experiments we applied SMAC v2 [29], which is one of the state-of-the-art configurators. It optimizes parameters for an arbitrary algorithm based on an instance set. In our case, the instance set consist of only one instance since we want to find best weights for each instance separately.

SMAC v2 [29] was used since the newer SMAC v3 [35] does not support constraining the parameter search yet. In our case this feature is crucial for the performance, otherwise SMAC would check equivalent parameter configurations multiple times. If SMAC is configured such that every weight parameter has an integer range between 0 and 10 then

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the search space has 10^3 possible configurations. However, many configurations lead to the same result. For example, the set of all configurations where all weight values are equal would lead to same schedule. Therefore, we added this constraint to avoid symmetric solutions: $w_{due\_date}, w_{critical\_ratio}, w_{setup\_time} = 10$. This constraint restricts the previously defined search space to 66 configurations. SMAC was configured such that it calls max. 15 times the weighted rule with different parameters.

SMAC needs a function to evaluate the performance of a parameter combination. The normalized multi-objective function based on the minimum and maximum of the grid search could be used. However, this would require executing the grid search which makes the automated algorithm configuration tool unnecessary. Therefore, we use the unnormalized objective function for SMAC.

3.4.5 Computational Environment

All experiments reported in the automated parameter configuration part of the thesis have been performed on a machine with an i5-8350U CPU and 16GB RAM. We implemented all machine learning approaches using the Keras framework from Chollet et al. [16] and the scikit-learn framework from Pedregosa et al. [46].

We generated a total of 4350 parallel machine scheduling instances by executing the random generator that has been described in Section 3.3.1 once for each possible combination of the following parameter sets:

$$\begin{aligned} P &\in \{1, 2, 4, \ldots, 9\} \\ J &\in \{10, 15, 20, \ldots, 150\} \\ M &\in \{10, 15, 20, \ldots, 150\} \text{ where } M \leq J \\ longerDurations &\in \{True, False\} \end{aligned}$$

80% of these instances were selected randomly and then used to train the machine learning models. The remaining 20% were used as the test set (small test set) to evaluate all approaches.

To analyze the scalability of the approaches, we further generated an additional test set (large test set) that consists of large scale instances using the following random instance generator parameter combinations:

$$\begin{aligned} P &\in \{10\} \\ J &\in \{460, 480, \ldots, 540\} \\ M &\in \{100\} \\ longerDurations &\in \{True, False\} \end{aligned}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

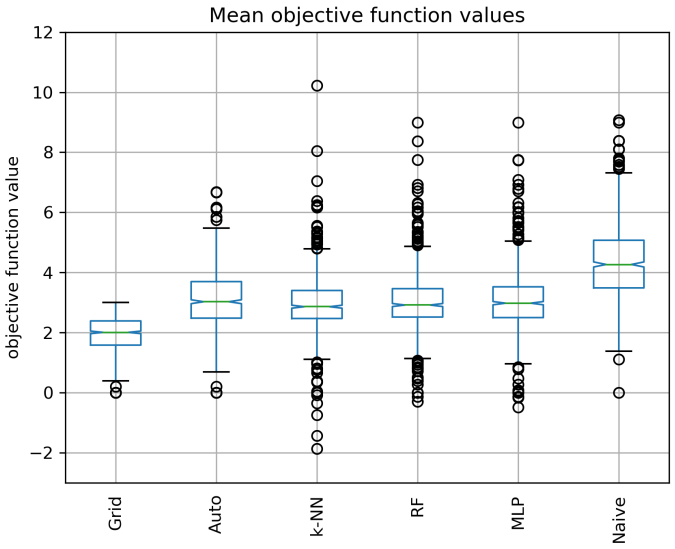


Figure 3.3: Mean objective function value on test instances

3.4.6 Computational Results

In this section we compare the results achieved by the machine learning based approaches to the results produced by other approaches that have been described in Section 3.4.4. We abbreviate all of the investigated approaches as follows: *Grid* stands for the exhaustive Grid search, *Auto* means the automated algorithm configurator SMAC, *k-NN* the machine learning approach k-nearest neighbor, *RF* the ML approach with random forest, *MLP* the ML approach with multi-layer perceptron and *Naive* is the approach with the default values.

We evaluate the performance of these approaches based on the achieved normalized objective values for each instance in the small test set. To obtain these values we executed the weighted rule using all the parameter configurations that have been determined by the different approaches. For each approach, the mean of all the corresponding normalized objective values was gathered. Figure 3.3 compares the mean values reached by the investigated approaches.

We can see that grid search yields better results compared to the other evaluated methods. SMAC with the non-normalized multi-objective function seems to have a similar performance to the machine learning approach in our experiments. However, these approaches require much more time and are not always useful for real-life scenarios, where optimized weights should be found quickly.

Our machine learning approach gives comparable results to SMAC, but requires much lower run times to determine good parameter configurations. The different ML models have a similar performance. All approaches lead to better results compared to the naive

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

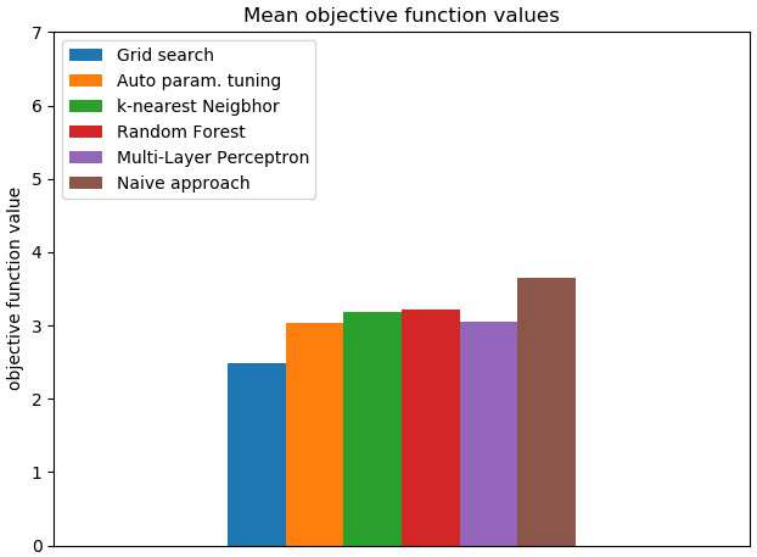


Figure 3.4: Mean objective function value on large instances

approach.

Table 3.1 and Figure 3.4 give more detailed information regarding the scalability of the approaches on the instances from the large test set.

	Grid	Auto.	k-NN	RF	MLP	Naive
time [min]	105.6	27.3	0.468	0.472	0.66	0

Table 3.1: Average run time required to determine a parameter configuration for instances from the large test set

As we mentioned before, we can see that exhaustive grid search and automated parameter tuning techniques are impracticable due to long running times. Comparing machine learning models on large instances, we can conclude that they achieve similar results, although slightly better results are obtained using the multi-layer perceptron model. The machine learning approaches produce comparable results to SMAC but need much less time to obtain a parameter configuration.

Figure 3.5 shows the absolute differences between the parameter configurations determined by each approach and the best configuration achieved by the exhaustive grid search. For each approach the aggregated mean absolute difference values for each of the three weight parameters over the whole test instance set is given.

We observe that the absolute mean difference obtained by SMAC is higher than the means reached by the other machine learning models. Figure 3.5 in combination with Figure 3.3

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

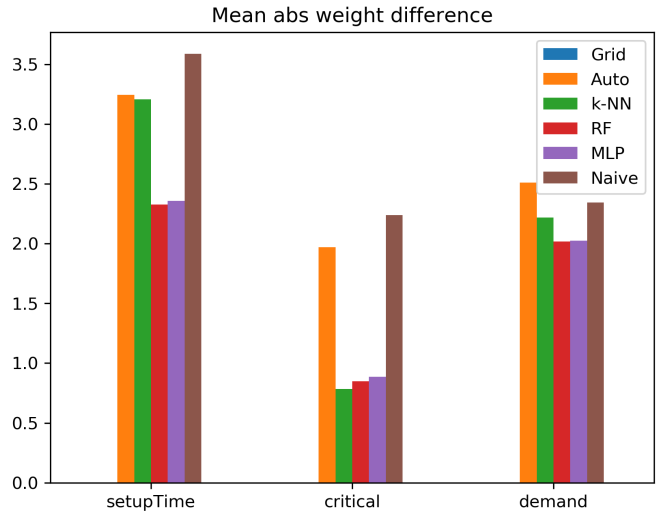


Figure 3.5: Mean absolute weight value difference:
Each approach has its own color and the bars are grouped by the weights. Compared to the grid search the used automated algorithm configuration tool identifies quite different parameter configurations.

indicates that SMAC and other machine learning models find parameter configurations in different search space regions compared to the grid search parameter configurations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Conclusion and Future Work

This work demonstrates the application of supervised machine learning to two arising problems in the industry. The prediction of the product quality is identified as a binary classification problem and the automated parameter configuration problem is tackled with multi-target regression.

Since our classification of the product quality is based on real-world data it recalls us of the importance to first analyze the provided data, filter out misleading information and afterwards extract useful features. We propose two feature sets that characterize products that are attached to a skid to predict the quality outcome of them. These feature sets are based on the production schedule of a paint shop. Many data preparation and machine learning techniques on labelled data sets are analyzed. This thesis also tries to overcome the issue of an unbalanced data set by exploring different countermeasures. A systematical workflow is used to select and evaluate a vast amount of machine learning approaches which consists of data preparation, the machine learning algorithm and optional post processing. For each step, one or more techniques can be selected where each technique can contain adjustable hyper parameters.

We are able to show that our best found machine learning approach performs better than a baseline classifier according to the considered performance metrics: accuracy, precision, recall, f1-score, confusion matrix and roc curve.

We also used the tool auto-sklearn to automatically find good machine learning models. This technique yields a model with high performance.

A recent and promising research is to understand complex machine learning models. The tool SHAP is used to determine the importance of features for machine learning models. This is helpful to understand *why* the model makes its predictions. Our experiments indicate that features related to frequent color changes have a great impact on the classifier predictions which coincides with the prior assumption about the negative effect of color changes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

At the automated parameter configuration part of the thesis, we propose a machine learning approach to automatically obtain instance specific parameter configurations for dispatching rules. We define novel features that are used to characterize parallel machine scheduling instances. With exhaustive grid search we obtain for each instance a ground truth parameter configuration and use a normalization scheme to encourage a fair comparison between different objectives. The final data set that is used for the multi target regression approach is based on the introduced features and the ground truth values determined by grid search.

A large number of experiments using unseen test instances show that dispatching rules that use the parameter configurations obtained by the machine learning approach produce high quality schedules. Furthermore, the results show that the proposed method requires much less run time to determine good parameter configurations than a state-of-the-art automated algorithm configuration method.

Therefore, we conclude that the machine learning approach can be used to obtain high quality parameter configurations for unseen instances of the parallel machine scheduling problem in very short run times. In future work, we plan to investigate analyze the importance of the individual features and plan to evaluate our approach on other problem domains.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] Ning An, Weigang Zhao, Jianzhou Wang, Duo Shang, and Erdong Zhao. Using multi-output feedforward neural network with empirical mode decomposition based signal filtering for electricity demand forecasting. *Energy*, 49:279–288, 2013.

[2] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, 2010.

[3] Yun Bai, Zhenzhong Sun, Jun Deng, Lin Li, Jianyu Long, and Chuan Li. Manufacturing quality prediction using intelligent learning approaches: A comparative study. *Sustainability (Switzerland)*, 10(1):85, 2017.

[4] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.

[5] John H. Blackstone, Don T. Phillips, and Gary L. Hogg. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, 20(1):27–45, 1982.

[6] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 5(5): 216–233, 2015.

[7] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation*, 20(1):110–124, Feb 2016.

[8] Jürgen Branke, Torsten Hildebrandt, and Bernd Scholz-Reiter. Hyper-heuristic evolution of dispatching rules: A comparison of rule representations. *Evolutionary Computation*, 23(2):249–277, 2015.

[9] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36(1-2):85–103, 1999.

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

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[11] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[12] Edmund K. Burke, Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: a survey of the state of the art. *JORS*, 64(12):1695–1724, 2013.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2(3):27:1–27:27, 2011.

[14] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16: 321–357, 2002.

[15] Wen-Chin Chen, Pei-Hao Tai, Min-Wen Wang, Wei-Jaw Deng, and Chen-Tai Chen. A neural network-based approach for dynamic quality prediction in a plastic injection molding process. *Expert Syst. Appl.*, 35(3):843–849, 2008.

[16] François Chollet et al. Keras. `https://keras.io`, 2015.

[17] Ahmed El-Bouri, Subramaniam Balakrishnan, and Neil Popplewell. Sequencing jobs on a single machine: A neural network approach. *European Journal of Operational Research*, 126(3):474–490, 2000.

[18] Charles Elkan. The foundations of cost-sensitive learning. In *IJCAI*, pages 973–978. Morgan Kaufmann, 2001.

[19] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *NIPS*, pages 2962–2970, 2015.

[20] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Practical automated machine learning for the automl challenge 2018. In *International Workshop on Automatic Machine Learning at ICML*, pages 1189–1232, 2018.

[21] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[22] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[23] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[24] Jacob Goldberger, Sam T. Roweis, Geoffrey E. Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *NIPS*, pages 513–520, 2004.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[25] Isabelle Guyon, Imad Chaabane, Hugo Jair Escalante, Sergio Escalera, Damir Jajetic, James Robert Lloyd, Núria Macià, Bisakha Ray, Lukasz Romaszko, Michèle Sebag, Alexander R. Statnikov, Sébastien Treguer, and Evelyne Viegas. A brief review of the chalearn automl challenge: Any-time any-dataset learning without human intervention. In *AutoML@ICML*, volume 64 of *JMLR Workshop and Conference Proceedings*, pages 21–30. JMLR.org, 2016.

[26] R. Haupt. A survey of priority rule-based scheduling. *Operations-Research-Spektrum*, 11(1):3–16, Mar 1989.

[27] Jens Heger, Jürgen Branke, Torsten Hildebrandt, and Bernd Scholz-Reiter. Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times. *International Journal of Production Research*, 54(22):6812–6824, 2016.

[28] Geoffrey E. Hinton. Connectionist learning procedures. *Artif. Intell.*, 40(1-3):185–234, 1989.

[29] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer, 2011.

[30] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer, 2011.

[31] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[32] Feng Ju, Jingshan Li, Guoxian Xiao, and Jorge Arinez. Quality flow model in automotive paint shops. *International Journal of Production Research*, 51(21): 6470–6483, Nov 2013.

[33] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[34] Paul Lange. Machine learning based error prediction for spray painting applications. Master's thesis, Chalmers University of Technology and University of Gothenburg, 2016.

[35] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp, and Frank Hutter. Smac v3: Algorithm configuration in python. `https://github.com/automl/SMAC3`, 2017.

[36] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Inf. Sci.*, 250:113–141, 2013.

[37] Gilles Louppe. Understanding random forests: From theory to practice, 2014.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[38] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017.

[39] Helmut Mausser. Normalization and other topics in multi-objective optimization. *Proceedings of the fields-MITACS Industrial Problems Workshop*, 2:89–101, Aug 2006.

[40] Mokotoff and Ethel. Parallel machine scheduling problems: A survey. *Asia - Pacific Journal of Operational Research*, 18(2):193–242, 11 2001.

[41] M. Montazeri and L. N. Van Wassenhove. Analysis of scheduling rules for an fms. *International Journal of Production Research*, 28(4):785–802, 1990.

[42] Maximilian Moser, Nysret Musliu, Andrea Schaerf, and Felix Winter. Solution methods for a real-life unrelated parallel machine scheduling problem. *Under submission*, 2019.

[43] Wiem Mouelhi-Chibani and Henri Pierreval. Training a neural network to select dispatching rules in real time. *Computers & Industrial Engineering*, 58(2):249–256, 2010.

[44] S. S. Panwalkar and Wafik Iskander. A survey of scheduling rules. *Operations Research*, 25(1):45–61, 1977.

[45] Andrew J. Parkes, Neema Beglou, and Ender Özcan. Learning the quality of dispatch heuristics generated by automated programming. In *LION*, volume 11353 of *Lecture Notes in Computer Science*, pages 154–158. Springer, 2018.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[47] Bernardete Ribeiro. Support vector machines for quality monitoring in a plastic injection molding process. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 35 (3):401–410, 2005.

[48] Juan Diego Rodríguez, Aritz Pérez Martínez, and José Antonio Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(3):569–575, 2010.

[49] I. Sabuncuoglu. A study of scheduling rules of flexible manufacturing systems: A simulation approach. *International Journal of Production Research*, 36(2):527–546, 1998.

[50] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[51] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[52] Donald F. Specht. A general regression neural network. *IEEE Trans. Neural Networks*, 2(6):568–576, 1991.

[53] Gerard V. Trunk. A problem of dimensionality: A simple example. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(3):306–307, 1979.

[54] Eva Vallada and Rubén Ruiz. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3):612–622, 2011.

[55] Felix Winter, Nysret Musliu, Emir Demirovic, and Christoph Mrkvicka. Solution approaches for an automotive paint shop scheduling problem. In *ICAPS*, pages 573–581. AAAI Press, 2019.

[56] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, jan 2016.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.