TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Verbesserung der Prognosegenauigkeit von Zielgrößen in der Produktionsplanung und -Steuerung unter der Anwendung von Supervised Machine Learning

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Diplom-Ingenieurs

unter der Leitung von

Priv.-Doz. Dr.-Ing. Fazel Ansari, M.Sc.
(E330 Institut für Managementwissenschaften, Bereich: Industrial Engineering, Forschungsgruppe Produktions- und Instandhaltungsmanagement)

Viola Gallina, PhD
(Fraunhofer Austria Reserch GmbH)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Johannes Glaeser, BSc

01426750

Blumauergasse 4

1020 Wien

Wien, im April 2023

_____
Johannes Glaeser

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Diplomarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre hiermit Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Wien, im April 2023

_____

Johannes Glaeser

Danksagung

Zu Beginn meiner Diplomarbeit möchte ich die Gelegenheit nutzen, um mich bei allen zu bedanken, die mich während meines Studiums unterstützt haben.

Zunächst möchte ich Herrn Priv.-Doz. Dr. Ing. Fazel Ansari und Frau Dr. Viola Gallina für die Betreuung meiner Diplomarbeit danken. Ohne ihre fachliche Expertise und ihre konstruktiven Anregungen wäre die Erstellung meiner Arbeit nicht möglich gewesen.

Ich möchte mich bei meinen Eltern und meinen Brüdern sowie ihren Familien für ihre stete Unterstützung während des gesamten Studiums bedanken. Ohne ihre Liebe, Fürsorge und Ermutigung wäre ich nicht so weit gekommen.

Besonders bedanken möchte ich mich auch bei Georg und Romana, die mir während der gesamten Zeit immer mit Rat und Tat zur Seite gestanden haben. Ohne ihre Unterstützung hätte ich viele Herausforderungen nicht gemeistert.

Zuletzt möchte ich mich bei all meinen Freunden für die unvergessliche Studienzeit bedanken. Die gemeinsamen Stunden im Hörsaal, bei Projekten und in der Freizeit waren für mich von unschätzbarem Wert und haben mich geprägt.

Kurzfassung

Die vierte industrielle Revolution stellt die Industrie vor nie dagewesene Herausforderungen. Steigende Kundenansprüche und hohe Liefertermintreue führen zur Zunahme der Komplexität in der Produktion. Dieser Umstand zwingt die Unternehmen zur Optimierung ihrer Produktionsplanung und -Steuerung (PPS). Traditionelle Modelle zur Vorhersage von Zielgrößen können die komplexen Zusammenhänge der Produktionsfaktoren oft nicht mehr erkennen und liefern keine akkuraten Ergebnisse.

Eine genaue Vorhersage von Zielgrößen ist jedoch für die Produktivität des Unternehmens entscheidend. Zu große Abweichungen zwischen der Produktionsplanung und der tatsächlichen Ausführung führen zu einem erhöhtem Koordinations- und Steuerungsaufwand. Dieser wirkt sich negativ auf den Bestand, die Durchlaufzeit und die Auslastung aus.

Um diesem Problem entgegenzuwirken, werden zunehmend Ansätze entwickelt, die maschinelle Lernalgorithmen zur Prognose von Zielgrößen in der Industrie einsetzen. Die Leistung der Algorithmen wurde bereits in einigen Studien erprobt und die Ergebnisse deuten auf ein großes Potential zur Verbesserung der Zuverlässigkeit in der PPS hin.

Im Zuge dieser Arbeit werden relevante Supervised Machine Learning (SML) Algorithmen ausgewählt und hinsichtlich ihrer Zuverlässigkeit zur Prognose von Zielgrößen in der PPS untersucht. Dazu werden die SML- Methoden anhand eines konkreten Anwendungsfall aus der Industrie getestet. Die Entwicklung der Modelle findet nach dem Cross Industry Standard Process for Data Mining (CRISP-DM) Modell statt. Die Vorbereitung des Datensatzes wird mithilfe von Feature Engineering durchgeführt.

Das Ergebnis dieser Arbeit ist ein Spektrum an SML- Modellen, die in der PPS zur Vorhersage von Zielgrößen eingesetzt werden können. Die angestrebte Reduktion der Abweichungen von mindestens 10 Prozent gegenüber dem Referenzmodell konnte im konkreten Anwendungsfall mithilfe einer Gradient Boosting Maschine (GBM) und der Anwendung von Feature Engineering und ohne Feature Auswahl erreicht werden. Alle übrigen getesteten SML- Algorithmen zeigen für den ausgewählten Anwendungsfall ebenfalls zufriedenstellende Ergebnisse.

Abstract

The fourth industrial revolution is presenting industry with unprecedented challenges. Increasing customer demands and high adherence to delivery dates lead to an increase in complexity in production. This circumstance forces companies to optimize their production planning and control (PPC). Traditional models for predicting target variables are often no longer able to recognize the complex interrelationships of the production factors and do not deliver accurate results.

However, accurate prediction of target variables is crucial for the company's productivity. Excessive deviations between production planning and actual execution lead to increased coordination and control efforts. This has a negative impact on inventory, lead time or capacity utilization.

To counteract this problem, approaches are increasingly being developed that use machine learning algorithms to predict target variables in industry. The performance of the algorithms has already been tested in some studies and the results indicate a great potential for improving reliability in PPC.

In the course of this work, relevant Supervised machine learning (SML) algorithms are selected and investigated with respect to their reliability for forecasting target variables in PPC. For this purpose, the SML methods are tested based on a concrete use case from industry. The development of the models takes place according to the Cross Industry Standard Process for Data Mining (CRISP-DM) model. The preparation of the data set is performed using feature engineering.

The result of this work is a spectrum of SML modells that can be used in PPS for the prediction of target variables. The targeted reduction of deviations of at least 10 percent compared to the reference model could be achieved in the specific use case with the help of a gradient boosting machine (GBM) and the application of feature engineering, but without feature selection. All other SML algorithms tested also show satisfactory results for the selected use case.

Inhaltsverzeichnis

1 Einleitung

1.1 Allgemeine Einleitung in das Themenfeld

Der beschleunigte technologische Wandel, der verstärkte Wettbewerb durch Globalisierung und die stetig steigenden Kundenansprüche stellen produzierende Unternehmen vor große Herausforderungen. Die vierte industrielle Revolution bringt mit ihren großen Potentialen auch eine immer höher werdende Komplexität in die Produktion mit sich. Lieferzeiten werden reduziert, die Variantenvielfalt nimmt zu und die Forderungen nach zuverlässigen Vorhersagen der Zielgrößen in der PPS werden lauter. (siehe Kapitel 2.1)

In der Vergangenheit wurden viele verschiedene Systeme zur Unterstützung der Planung und Steuerung von Zielgrößen in der Produktion etabliert. Jedoch konnten die Daten oft nicht in der benötigten Zeit und Qualität zur Verfügung gestellt werden. Neue Systeme in der Industrie 4.0, wie Cyber-Physische Produktionssysteme, ermöglichen heutzutage den Abruf qualitativer Daten der Produkte, Menschen und Prozesse. Dadurch werden neue Potentiale zur Steigerung der Wettbewerbsfähigkeit für die PPS geschaffen, die bis heute noch nicht ausgeschöpft werden.[1]

1.2 Problemstellung, Ziele und Forschungsfragen

1.2.1 Problemstellung

Die genaue Vorhersage von Zielgrößen spielt eine entscheidende Rolle für die Qualität und Effizienz der PPS. Traditionelle PPS- Modelle berechnen oft durchschnittliche Werte aus historischen Daten, um Zielgrößen für die Produktion vorherzusagen. Diese liefern jedoch keine akkuraten Ergebnisse, da sie die Zusammenhänge mehrerer Einflussfaktoren in der komplexen Produktionsumgebung nicht erkennen können.[2]

Schuh et al. konnten in ihrer Studie feststellen, dass die Zuverlässigkeit der Produktionsplanung und damit die Planungsqualität, bezogen auf den Planungshorizont eines deutschen mittelständischen Maschinenbauunternehmen, in den ersten drei Tagen nach der Planung auf 25 Prozent sinken kann. Für den Durchlauf eines Produktes wird allerdings oft eine zuverlässige Vorhersagegenauigkeit von mehreren Wochen oder Monaten benötigt. Diese Umstände führen zu

[1] vgl. Ryback et al., 2019, S.131
[2] vgl. Lingitz et al., 2018

Verschwendungen wie unnötigen Rüstvorgängen, Maschinen-Wartezeiten und Eilaufträgen, wodurch die Produktivität des Unternehmens gesenkt wird.[3]

Ansätze, die ausschließlich auf maschinellem Lernen basieren, weisen ein großes Potenzial auf, um die Probleme traditioneller PPS-Modelle zu überwinden und die Zuverlässigkeit in der PPS zu erhöhen. In den vergangenen Jahren wurden dazu erst wenige Studien publiziert, die den Einsatz von Machine Learning in der PPS hinsichtlich ihrer Planungsqualität untersuchen. Zudem gibt es bis heute keine offensichtliche Definition der Planungsqualität.[4]

Die spezifischen Produktionsplanungsprobleme (P) ergeben sich wie folgt:

P 1: Traditionelle Vorhersagemodelle liefern hinsichtlich der Zuverlässigkeit der Vorhersage von Zielgrößen keine zufriedenstellenden Ergebnisse. Dies liegt zum einen an der Komplexität in der Produktion, zum anderen an den nicht erkannten Zusammenhängen von Produktionsfaktoren.

P 2: Die vorhergesagten Prozesszeiten bei der Herstellung von Blechen eines bekannten österreichischen Stahlherstellers weichen stark von den ermittelten Planzeiten aus dem Enterprise Ressource Planning (ERP) System ab. Die Herausforderung besteht darin, ein Modell zu entwickeln, das die Prognosegenauigkeit der Prozesszeiten verbessert.

1.2.2 Ziele und Forschungsfragen

Übergeordnete Forschungsfrage:

Wie kann unter der Anwendung von Supervised Machine Learning (SML) die Prognosegenauigkeit von Zielgrößen in der PPS verbessert werden?

Ziele der Arbeit:

Das Ziel dieser Arbeit ist es, zunächst einen Überblick über den Einsatz von SML-Algorithmen zur Verbesserung der Prognosegenauigkeit in der PPS zu schaffen. Die Erkenntnisse daraus sollen genutzt werden, um die Vorhersagezuverlässigkeit der PPS und damit die Planungsqualität anhand eines konkreten Produktionsplanungsproblems aus der Industrie – genauer aus der Stahlindustrie - zu verbessern. Dafür sollen einige SML – Algorithmen anhand der Häufigkeit der Verwendung in der Literatur ausgewählt und hinsichtlich ihrer Planungsqualität miteinander verglichen werden:

Eine höhere Planungsqualität bedeutet, dass die Abweichungen der prognostizierten Prozesszeiten von den tatsächlichen Prozesszeiten wirtschaftlich und betrieblich in

[3] vgl. Schuh et al., 2015, S.6f
[4] vgl. Ryback et al., 2019, S.131

einem vertretbaren Rahmen bleiben. Als Referenzmodell wird ein bereits entwickelter Random Forest Ansatz herangezogen, der ohne Feature Engineering entwickelt wurde.

Als Ziel strebt diese Arbeit eine Reduktion der Abweichungen von mindestens 10 Prozent gegenüber den Abweichungen des bereits entwickelten Vorhersagemodells an. Als Ground Truth werden dafür die vom Unternehmen gestellten Prozessdaten herangezogen, welche die tatsächlichen Prozesszeiten beinhalten. Die Stammdaten werden, im Vergleich zum Referenzmodell, mithilfe von Feature Engineering für die Modelle vorbereitet.

Erwartetes Ergebnis der Arbeit:

Das Ergebnis dieser Arbeit wird ein Spektrum an ausgewählten SML- Algorithmen, die in der PPS am häufigsten zur Prognose von Zielgrößen eingesetzt werden können. Mit dieser Arbeit werden folgende Forschungsfragen beantwortet:

F1) Welche SML-Methoden zur Prognose von Zielgrößen in der PPS werden in der Literatur beschrieben?

F2) Wie hoch ist die Zuverlässigkeit ausgewählter SML- Algorithmen für die Prognose von Zielgrößen in der PPS und speziell in der Stahlindustrie?

1.3 Methodisches Vorgehen

Im folgenden Unterkapitel wird im genaueren auf die ausgewählte und verwendete Methode eingegangen, die dieser Arbeit zu Grunde liegt. Der theoretische Hintergrund, sowie der aktuelle Stand der Technik wurden mithilfe einer Literatur- und Internetrecherche ermittelt. Die genaue Methode zur Erhebung der Studien zum aktuellen Forschungstand ist Kapitel 3.1 zu entnehmen.

Um die Prognosegenauigkeit der SML- Algorithmen zu verbessern, ist es notwendig eine standardisierte und strukturierte Vorgehensweise zu wählen, die sich wiederholen und nachvollziehen lässt. Dazu bietet sich das viel genutzte *Cross Industry Standard Process for Data Mining (CRISP-DM)* Modell an. Dieses Modell beinhaltet die sechs Schritte *Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation* und *Deployment*.[5]

Diese Arbeit konzentriert sich in erster Linie auf die Schritte *Data Preparation*, *Modeling* und *Evaluation.* In der Phase *Data Preparation* wird der Datensatz mithilfe des Feature Engineering für das jeweilige Modell vorbereitet. In der Phase *Modeling* werden die Modellierungstechniken (SML-Algorithmen) ausgewählt und mit den

[5] vgl. Wirth & Hipp, 2000

gegebenen Daten antrainiert. In der Phase *Evaluation* werden die Modelle hinsichtlich ihrer Leistung untersucht und mithilfe von Testdaten getestet.

Das **CRISP-DM Modell** stammt ursprünglich aus dem *Data Mining*, wird aber mittlerweile sehr viel in der Machine-Learning-Community in Analyseprojekten jeglicher Art eingesetzt.[6]

Das Prozessmodell gibt einen Überblick über den Lebenszyklus eines Data-Mining-Projekts. Der Lebenszyklus besteht dabei aus sechs Phasen, wie sie in Abbildung 1 abgebildet sind. Die einzelnen Phasen beinhalten die Aufgaben des Projekts und durch die Pfeile werden die Beziehung zwischen den Phasen gekennzeichnet. Die Pfeile zeigen jedoch nicht alle Beziehungen, die zwischen den Phasen bestehen, sondern deuten auf die am häufigsten zu beobachtenden Sprünge hin. Je nach Datenlage und Erkenntnisse innerhalb des Zyklus werden manche Stationen öfters durchlaufen, aber auch Phasen ausgelassen oder zeitweise übersprungen. Der äußere Kreis symbolisiert die zyklische Anwendung des Lebenszyklus im Unternehmen. Die aus den Prozessen gewonnen Erkenntnisse sollen für weitere Geschäftsfragen und Folgeprojekten genutzt werden. [7] Die genannten sechs Phasen werden nun überblicksmäßig beschrieben. Für eine genauere Beschreibung der Phasen wird auf Chapman et al. verwiesen.

Mit dem **Business Understanding** (Business Verständnis) wird der CRISP- DM Prozessmodell eingeleitet. Der erste Schritt besteht darin, den Geschäftskontext sowie die Rahmenbedingungen des Anwendungsszenarios zu verstehen. Weiters werden die geschäftlichen Anforderungen und Ziele daraus abgeleitet. Die Ziele müssen messbar sein, da es sonst schwierig wird die Ergebnisse der Analyse zu bewerten.[8]

In der Phase **Data Understanding** (Verständnis der Daten) werden die zur Verfügung stehenden Daten erstmals ausgemacht. Die für die Analyse relevanten und geeigneten Datenquellen werden identifiziert und erste Erkenntnisse für den weiteren Verlauf gewonnen. In dieser Phase sollen bereits erste Probleme, wie mangelnde Datenqualität oder die Menge und Qualität der Daten untersucht werden. Für etwaige Mängel müssen geeignete Maßnahmen zur Verbesserung der Datenqualität getroffen werden. [9]

[6] vgl. Schacht & Lanquillon, 2019, S. 112 f
[7] vgl. Chapman et al., 2000, S. 13 ff
[8] vgl. Schacht & Lanquillon, 2019, S. 113 f
[9] vgl. Schacht & Lanquillon, 2019, S. 114

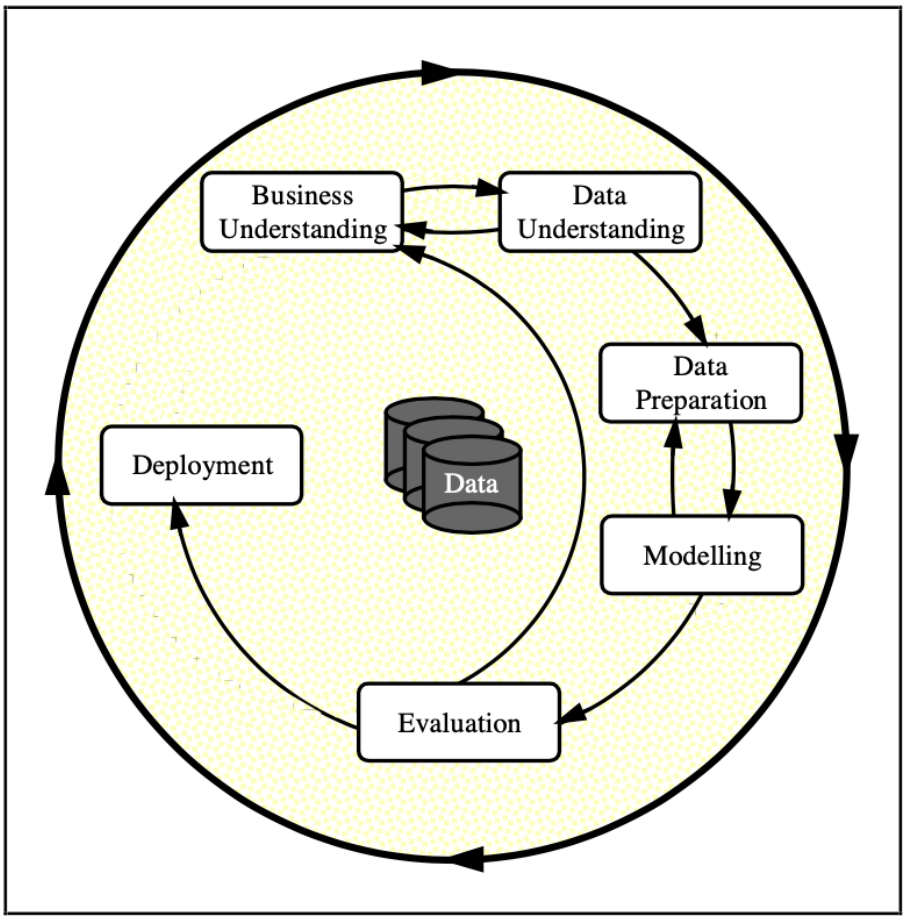

Abbildung 1: Phasen des CRISP-DM-Prozessmodells für Data Mining[10]

Bei der dritten Phase **Data Preparation** (Datenvorbereitung) handelt es sich um die zeitaufwändigste Phase im gesamten Prozesszyklus. In der Regel werden für diese Phase etwa 60 bis 70 Prozent der Zeit des Gesamtprojekts aufgewendet. Der Grund dafür liegt darin, dass schlecht aufbereitete Daten auch zu schlechten Modellen mit geringer Leistung führen. Daher wird diese Phase in der Literatur als die Wichtigste der sechs Phasen bezeichnet. Zu den Hauptaufgaben der Phase Data Preparation zählt das Bereinigen, das Ordnen, das Kuratieren und Vorbereiten der Daten für das ausgewählte Modell. In dieser Phase werden je nach Methode etwa fehlende Daten ergänzt oder Ausreißer -Daten entfernt. Außerdem findet in dieser Phase auch die Auswahl der richtigen Attribute (Features) statt.[11]

Die Phase **Modelling** (Modellierung) beinhaltet die Auswahl der Modellierungstechnik, den Aufbau des Testfalls und die Konstruktion des Modells. Im Allgemeinen kann jedes Data-Mining-Tool verwendet werden, sofern die Wahl begründet ist. Denn bei der Auswahl muss das Geschäftsproblem sowie die Art der Daten berücksichtigt werden. Das Modell muss mit bestimmten Parametern erstellt werden, um ordnungsgemäß zu

[10] Wirth und Hipp, 2000, S. 5
[11] vgl. Sarkar et al., 2018, S. 50

funktionieren. Um das Modell zu bewerten, wird empfohlen, das Modell anhand von Bewertungskriterien zu bewerten und danach das Geeignetste auszuwählen.[12]

Bevor das erstelle Modell endgültig im Unternehmen eingesetzt werden kann, muss das Modell hinsichtlich seiner Qualität gründlich analysiert und bewertet werden. In der Phase **Evaluation** (Evaluierung) wird überprüft, inwieweit das Modell die Unternehmensziele erfüllen kann. Dazu zählen auch Aspekte aus der Qualitätssicherung, wie etwa die Sicherstellung der korrekten Erstellung des Modells oder die Überprüfung der ausgewählten Attribute. Zur Durchführung der Überprüfung kann das Modell bereits anhand realer Anwendungen getestet werden, sofern dies die Zeit- und Budgetbeschränkungen des Unternehmens zulassen. In dieser Phase wird auch entschieden, ob das Projekt an dieser Stelle als abgeschlossen gilt, oder ob weitere Iterationsschritte und Sprünge in vergangene Phase notwendig sind.[13]

Die letzte Phase **Deployment** (Einsatz) beschreibt die Einführungsphase des Modells im Unternehmen. Diese Phase wird in vielen Fällen bereits vom Kunden selbst durchgeführt. Dazu muss der Kunde die notwendigen Maßnahmen ergreifen, um das Modell tatsächlich nutzen zu können. Eine richtige Kommunikation zwischen dem Ersteller des Modells und dem Kunden ist hier entscheidend.[14]

1.4 Aufbau und Struktur der Arbeit

In einem ersten Schritt wird der für die Arbeit wichtige theoretische Hintergrund bezüglich der PPS sowie des maschinellen Lernens geklärt. Außerdem wird der Begriff des Feature Engineering definiert und die wesentlichen Arbeitsabläufe des Feature Engineering vorgestellt.

Im nächsten Schritt wird ein Überblick über den aktuellen Stand der Technik (State of the Art) bezüglich des Einsatzes von Machine Learning in der PPS gegeben. Dafür werden einige relevante Studien herangezogen und genauer betrachtet. Aus den Erkenntnissen soll abgeleitet werden, welche SML-Algorithmen in der PPS bisher verwendet werden und welche davon die besten Ergebnisse liefern können.

Anschließend erfolgt die Auswahl geeigneter SML- Algorithmen aus der Literatur. Cadavid et al. gibt eine systematische Übersicht über veröffentlichte Studien zum Thema Machine Learning in der PPS. Die drei meistverwendeten SML - Algorithmen, die in PPS verwendet werden, sind laut ihrer Recherche Entscheidungsbaummodelle, Neuronale Netzwerke und Support Vector Machines. Unter Berücksichtigung der Datensätze, die vorwiegend numerische Werte enthalten, werden für diese Arbeit Regressionsmodelle verwendet. Anhand der Häufigkeit der Verwendung in der

[12] vgl. Schröer et al., 2021, S. 527
[13] vgl. Shearer, 2000, S. 18
[14] vgl. Chapman et al., 2000, S. 14

Literatur nach Cadavid et al., sowie den Erkenntnissen aus der Recherche zum Stand der Technik aus Kapitel 3, werden dafür folgende SML-Algorithmen genauer untersucht und verwendet: Ein Lineares Regressionsmodell, drei Entscheidungsbaummodelle, ein Neuronales Netzwerk sowie die Methode der Support Vector Machine. Diese Modelle werden in der Phase *Modelling* mithilfe der Scikit- Learn Bibliothek von Python programmiert.

Unter der Zuhilfenahme von Feature Engineering soll zunächst in der Phase Data Preparation die Prognosegenauigkeit weiter verbessert werden. Die Entwicklung und Auswahl von Features ist kein einmaliger Prozess, sondern findet, im Rahmen des CRISP – DM-Prinzip, iterativ statt. In der Datenwissenschaft wird die Aufgabe der richtigen Auswahl von Features als den schwierigsten Schritt beim Aufbau eines geeigneten Systems für maschinelles Lernen betrachtet. Sie hat demnach auch einen sehr großen Einfluss auf die auf die Leistung des Prognosemodells. Das Feature Engineering sorgt im Wesentlichen für eine bessere Leistung des Modells, sowie einer Minimierung der Berechnungszeit.

Mithilfe des Feature Engineerings werden die Datensätze für die jeweiligen SML-Algorithmen vorbereitet. Der Arbeitsablauf im Feature Engineering erfolgt in dieser Arbeit in zwei Hauptschritten:

1) **Feature extraction and engineering:** Dazu gehören der Umgang mit fehlerhaften Daten, das Einfügen fehlender Werte oder die Umwandlung bestimmter Werte.

2) **Feature selection:** Das Ziel der Feature selection besteht darin, eine optimale Anzahl an Features für das Training des Modells auszuwählen.

Für die Datenvorbereitung wird die Bibliothek *Feature Engine* von Python herangezogen.

Die antrainierten Machine Learning Algortihmen werden weiters hinsichtlich ihrer Leistung untersucht. Der gegebene Datensatz enthält historische Daten über die gesamten Prozesse innerhalb eines Kalenderjahres. Die tatsächlich gemessenen Prozesszeiten sind bereits im Vorfeld ermittelt worden und somit bekannt. Die Genauigkeit der Modelle wird nach Lingitz et al. mit fünf verschiedenen Fehlermaßstäben gemessen werden. Mithilfe der Formeln für die Standardabweichung wird die Planungsqualität ermittelt und visuell dargestellt. Zuletzt werden die Ergebnisse einander gegenübergestellt und miteinander verglichen werden. Für den Vergleich dienen die zuvor ermittelten Kennzahlen.

2 Theoretische Grundlagen

Dieses Kapitel behandelt die wichtigsten theoretischen Grundlagen, die für das Verständnis dieser Arbeit von Bedeutung sind. Zunächst wird der Begriff der PPS geklärt und ihre wesentlichen Ziele beschrieben. Als nächstes wird die Theorie zum maschinellen Lernen erläutert, wobei der Fokus auf der überwachten Lernmethode liegt. Zuletzt wird genauer auf die Datenvorbereitung eingegangen und wichtige Begriffe aus dem Feature Engineering geklärt.

2.1 PPS

Die produzierende Industrie unterliegt seit Jahren einem strukturellen Wandel. Individuelle Fertigungsaufträge und ein steigender Preisdruck bringen höhere Komplexität in die Produktion und zwingen die Unternehmen zur Optimierung der PPS. Dabei wurde der Begriff der PPS Anfang der 1980er Jahre zum ersten Mal eingeführt, um in der produzierenden Industrie die Material- und Zeitwirtschaft in ein übergeordnetes Konzept zu umfassen. Der Begriff hat sich im Laufe der Zeit sowohl auf unternehmerischer Ebene als auch in der akademischen Forschung etabliert. Vielfach wird auch der Begriff *Enterprise Resource Planning (ERP*) als Synonym verwendet, der als Überbegriff für die Planung der Ressourcen und Produktionsprozesse verwendet wird.[15]

Die PPS macht sich zur Aufgabe, die Vorausplanung des laufenden Produktionsprogramms, trotz unvorhersehbarer Ereignisse, wie Maschinenstörungen oder Personalausfälle, für mehrere Perioden im Voraus zu gewährleisten. Das im deutschsprachigen Raum bekannte **Aachener PPS – Modell** unterteilt die Aufgaben der PPS in die Kernaufgaben, sowie die Querschnittsaufgaben. (siehe Abbildung 2)

Die Kernaufgaben der PPS umfassen die Produktionsprogrammplanung, die Produktionsbedarfsplanung sowie die Planung und Steuerung von Fremdbezug und Eigenfertigung. In der *Produktionsprogrammplanung* wird die Menge und Reihenfolge der zu produzierenden Erzeugnisse festgelegt. Die *Produktionsbedarfsplanung* leitet daraus den erforderlichen Material- und Ressourcenbedarf ab. In der *Eigenfertigungsplanung und -steuerung* werden Losgrößen berechnet, die Reihenfolge der Produktion geplant oder Verfügbarkeiten überprüft.[16]

[15] vgl. Schuh, 2007, S. 3 f
[16] vgl Lödding, 2005, S. 7

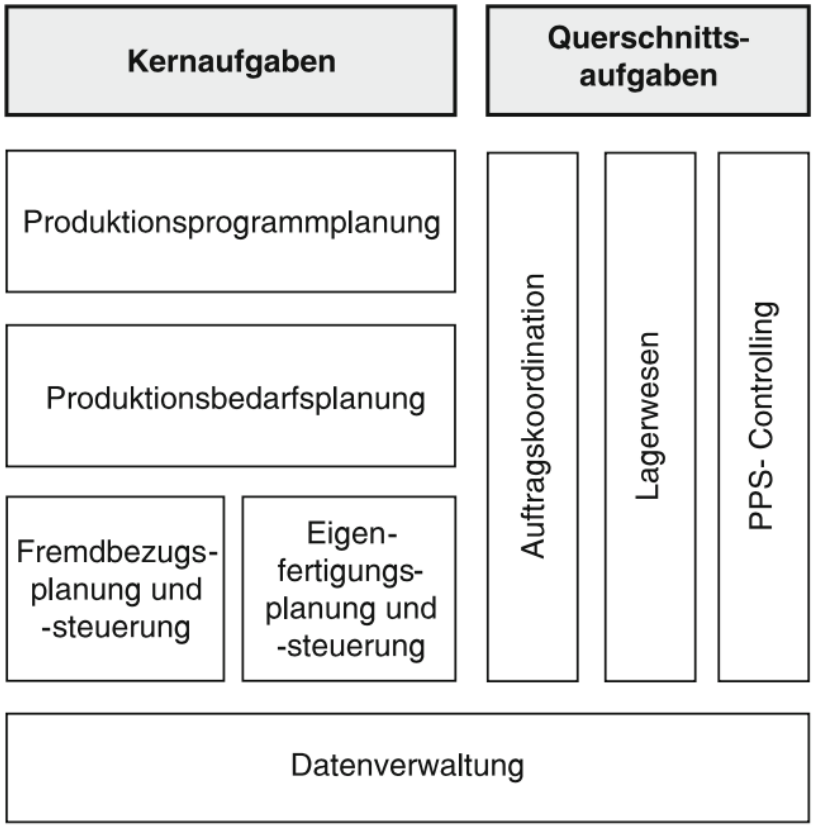

Abbildung 2: Aufgaben in der PPS nach dem Aachener PPS- Modell[17]

Die Querschnittsaufgaben dienen der Integration und Optimierung der PPS. Zu den Aufgaben zählen das Auftragsmanagement, das Bestandsmanagement und das PPS-Controlling. Im *Auftragsmanagement* sind die Aufgaben der Auftragsplanung,-steuerung und -überwachung zusammengefasst. Das Ziel des *Bestandsmanagements* ist, das zur Verfügung stellen der richtigen Bestände in der entsprechenden Menge zum richtigen Zeitpunkt.[18] Das PPS - Controlling dient zur Überwachung des Auftragsfortschrittes, dem Erkennen möglicher Störungen, sowie dem Setzten geeigneter Maßnahmen zur Verbesserung der Leistungsfähigkeit und Reduzierung der Kosten.[19]

[17] Lödding, 2005, S. 6
[18] vgl. Schuh, 2007, S. 58ff
[19] vgl. Schmidt und Nyhuis, 2021, S. 162f

2.2 Machine Learning

2.2.1 Definition

Der Begriff *Machine Learning*, stammt aus dem Bereich der *künstlichen Intelligenz (KI)*. Im Englischen wird der Begriff *Artificial Intelligence (AI)* verwendet. Der Begriff der KI wird bereits in den 50er Jahren von McCarthy et al.[20] definiert und beschreibt Maschinen, die sich mit einer Art der menschlichen Intelligenz verhalten. In den darauffolgenden Jahren hat sich aus der künstlichen Intelligenz das Teilgebiet des Maschinellen Lernen herausgebildet.[21]

Tom Mitchell definierte Maschinelles Lernen später, im Jahre 1997 folgendermaßen: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance a tasks in T, as measured by P, improves with experience E."* [22]

Diese Definition kann wie folgt vereinfacht werden: Machinelles Lernen ist ein Bereich, der aus Lernalgorithmen besteht, die Ihre Leistung P über die Zeit durch die Erfahrung E verbessern, beim Ausführen einer Aufgabe T.[23]

Allgemein kann Machine Learning als eine computergestützte Methode definiert werden, die mithilfe von Erfahrungen seine Leistung verbessert oder Vorhersagen treffen kann. Dafür müssen dem Modell zunächst Informationen aus der Vergangenheit übermittelt und für die Analyse zur Verfügung gestellt werden. In den meisten Fällen handelt es sich dabei um vom Menschen erstellte digitalisierte Trainingssätze oder durch Interaktion mit der Umgebung gewonnene Daten. Von diesen Erfahrungen lernt das System mithilfe von Algorithmen die Daten zu verstehen und entwickelt ein Vorhersagekonzept für die Zukunft. Der Umfang sowie die Qualität der Daten spielen die entscheidende Rolle für den Erfolg des Machine Learning Modells. Daher ist maschinelles Lernen sehr eng mit Datenanalyse und der Statistik verbunden.[24]

Machine Learning wird in der Zwischenzeit in einer Vielzahl von Bereichen eingesetzt. Neben den bekannten Anwendungen wie der Spamerkennung von Emails oder der Erkennung von Bildern, findet sich der Einsatz von Machine Learning Algorithmen auch in anderen Disziplinen wieder. Dazu zählen unter anderem die Medizin, Wissenschaft und Biologie, aber auch in der Rechtsprechung kommen die Algorithmen mittlerweile zum Einsatz. Ein bekanntes Beispiel ist die Wettervorhersage, bei der versucht wird mithilfe einer großen Anzahl an Eingangsparametern durch ein

[20] vgl. McCarty et al., 1955
[21] vgl. Welsch et al., 2018, S. 370
[22] Mitchell, 1997, S. 1
[23] vgl. Sarkar et al., 2018, S. 10
[24] vgl. Mohri et al., 2018, S. 1

Gleichungssystem die Wetterbedingungen vorherzusagen. Wenn die Möglichkeiten der herkömmlichen Lösungsmethoden ausgeschöpft sind, bietet das maschinelle Lernverfahren eine leistungsstarke Alternative zur Problemlösung, indem es aus den Vergangenheitsdaten lernt und mögliche, zuvor möglicherweise unbekannte Zusammenhänge, in der Datenstruktur findet. Mithilfe der Algorithmen können aber nicht nur Vorhersagen getroffen, sondern auch Einblicke in der Vorhersagestruktur der Daten gewonnen werden.[25]

2.2.2 Begriffsabgrenzung

Wie bereits erwähnt, wird das Maschinelle Lernen als ein Teilbereich der künstlichen Intelligenz betrachtet. In der Literatur werden Begriffe wie *deep learning*, *data mining* oder *data science* im Zusammenhang mit *machine learning* verwendet und oft fälschlicherweise damit verwechselt. Dabei handelt es sich beim maschinellen Lernen um einen multidisziplinären bzw. interdisziplinären Bereich, der mit vielen anderen Bereichen aus der Informatik interagiert. Abbildung 2 bietet einen guten Überblick über die wichtigsten Bereiche, die sich mit dem maschinellen Lernen im Hinblick auf die Konzepte, Methologien, Ideen und Techniken überschneiden.[26]

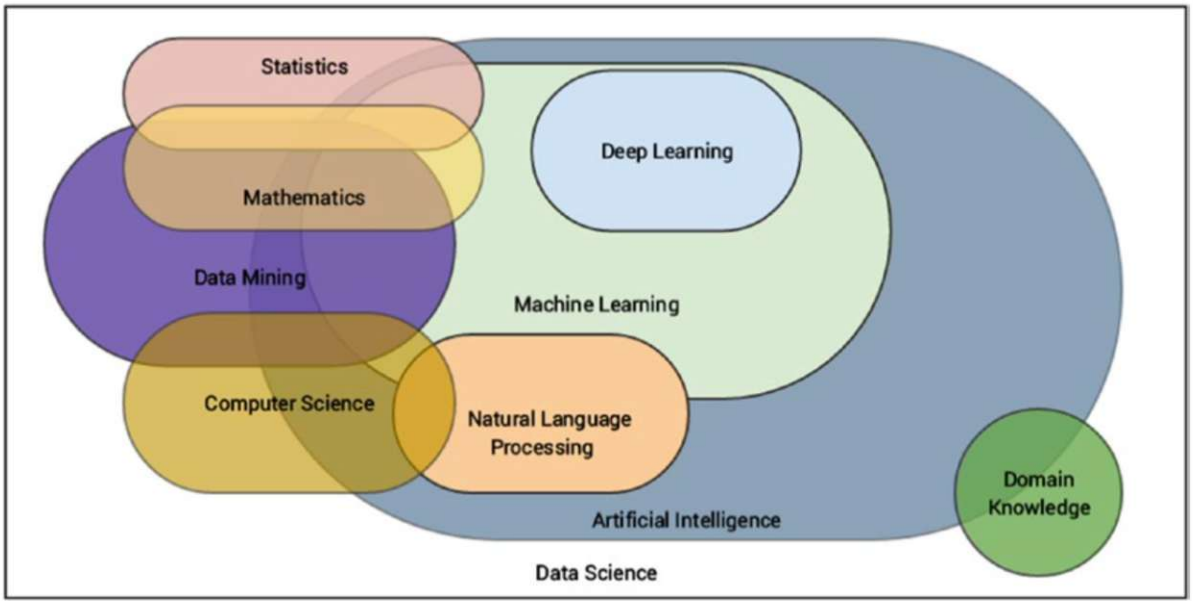

Abbildung 3: Multidisziplinarität des Begriffs Machine Learning[27]

Die **Künstliche Intelligenz** bildet die Grundmenge und umfasst das maschinelle Lernen als einen ihrer Spezialbereiche. Der Kerngedanke der künstlichen Intelligenz umfasst die Erforschung und Entwicklung der Intelligenz von Maschinen. Sie stützt sich auf Bereiche aus der Mathematik, der Statistik, der Informatik, den kognitiven

[25] vgl. Louppe, 2014, S. 1
[26] vgl. Sarkar et al., 2018, S. 13
[27] Sarkar et al., 2018, S. 13

Wissenschaften, der Linguistik, den Neurowissenschaften und vielen anderen Disziplinen.

Deep Learning ist ein Teilbereich des Machine Learnings und befasst sich mit Techniken des repräsentativen Lernens. Beim Deep Learning wird, im Unterschied zum Machine Learning, das Modell durch stetig neue Daten und Erfahrungen, meist mithilfe Neuronaler Netzwerke, verbessert.[28]

Data Mining, oft auch als Knowledge Discovery in Data- Bases (KDD) bezeichnet, ist eine Teildisziplin der Informatik und beschäftigt sich mit der automatischen Auswertung großer Datenmengen, die durch die neuen Technologien produziert werden. Mithilfe von Data Mining werden nützliche Muster (patterns) in den Daten erkannt, die für die Problemstellung relevant sind.[29]

2.2.3 Lernmethoden im Machine Learning

ML- Algorithmen sind mittlerweile allgegenwärtig und werden in vielen unterschiedlichen Bereichen eingesetzt. Insbesondere dort, wo Probleme durch große Datenmengen gelöst werden sollen, unterstützt das maschinelle Lernen die Tätigkeiten der Menschen. Um verstehen zu können welche Probleme durch die Anwendung von maschinellem Lernen behandelt werden können, sollen zunächst die gängigen Lernformen im maschinellen Lernen beschrieben werden. So unterscheidet man zwischen *dem Supervised Learning*, dem *Unsupervised Learning*, einer Mischform *Semisupervised Learning*, sowie *Reinforcement Learning*.[30] In der Literatur werden außerdem noch weitere Lernmethoden genannt, die für die weitere Arbeit jedoch keine relevante Rolle spielen.

Supervised Learning (zu deutsch: *überwachtes Lernen*) wird auch als das *Lernen aus Beispielen* bezeichnet. Anhand von Beispieldaten, die im Vorfeld bekannt sind, wird ein Modell antrainiert. Dieses Modell soll eine Funktion ermitteln, die für die gegebenen Eingangswerte die bekannten Ausgabewerte abbildet. Mithilfe des angelernten Modells sollen für neue Eingabewerte sinnvolle Ausgabewerte ermittelt werden. Da die korrekten Zielwerte bekannt sind, kann die Korrektheit des Modells und die Qualität des Lernvorgangs überwacht werden, woher der Begriff der *Überwachung* stammt.[31]

Mit Methoden des Supervised Learning können Problemstellungen, wie die Erkennung einer Handschrift, die Vorhersage von Aktienpreisen oder die Erkennung einer Sprache gelöst werden. Als Eingabewerte werden beispielsweise die Unternehmensdaten herangezogen, während der Ausgangswert der Preis der Aktie

[28] vgl. Sarkar et al., 2018, S.14
[29] vgl. Kriegel et al., 2007, S. 87
[30] vgl. Schacht & Lanquillon, 2019, S. 95
[31] vgl. Schacht & Lanquillon, 2019, S. 96

darstellt. Der Eingabevektor wird dabei als *Featurevektor*, der Ausgabevektor als *Label* bezeichnet. Die wichtigsten Aufgabentypen stellen die *Klassifikation* und *Regression* dar. Diese beiden Aufgabentypen werden im weiteren Verlauf der Arbeit noch genauer beschrieben.[32]

Unsupervised Learning, das *unüberwachte Lernen*, wird auch als das *Lernen aus Beobachtungen* bezeichnet. Im Gegensatz zum supervised Learning, erhält das Modell *ungelabelte* Trainingsdaten als Eingabewerte, allerdings keine Zielgrößen als Ausgabewerte. Die geringeren Vorgaben erleichtern die Handhabung mit dieser Lernmethode, da die oft aufwendige Erarbeitung der Labels entfällt. Der Begriff des *überwachten* Lernens rührt daher, dass für die Lösung des Modells nicht als Richtig oder Falsch bezeichnet werden kann und durch das Fehlen der Zielwerte über die Güte des Modells keine Aussagen getroffen werden kann.[33] Das Ziel des unüberwachten Lernens ist es, aus den ungelabelten Trainingsdaten Muster oder Strukturen zu erkennen und die Grundlage für weitere Datenanalysen zu schaffen. Die am häufigsten erforschte und angewandte unüberwachte Lerntechnik ist das *Clustering*. [34]

Semisupervised Learning (zu deutsch: *halbüberwachtes Lernen*) kombiniert Elemente aus dem überwachten und unüberwachtem Lernen. Dem Modell werden sowohl gelabelte als auch ungelabelte Trainingsdaten zur Verfügung gestellt. Man erhofft sich durch die Kombination der beiden Lerntypen eine verbesserte Leistung des Modells erreichen zu können. Halbüberwachtes Lernen wird oft für Problemstellungen herangezogen, in denen einerseits die Beschaffung von gelabelten Trainingsdaten (Zielgrößen), zwar möglich, aber teuer ist und andererseits ungelabelten Daten leicht zur Verfügung gestellt werden können.[35]

Reinforcement Learning, das *bestärkende Lernen*, unterscheidet sich grundlegend von den anderen genannten Lerntypen. Dem Lernalgorithmus werden im Vergleich keine Trainingsdaten zugespielt. Die Daten entstehen erst durch die Interaktion mit der Umgebung. Durch einen Prozess von Versuch und Irrtum, sowie Belohnung und Bestrafung (negative Belohnung) werden Daten generiert und das Modell lernt die Belohnungen zu maximieren. Als Beispiel kann ein neuronales Netzwerk durch den Einsatz von bestärkendem Lernen gute Spielstrategien für das Brettspiel Backgammon entwickeln.[36]

[32] vgl. Richter, 2019, S. 1
[33] vgl. Schacht & Lanquillon, 2019, S. 97
[34] vgl. Zhou, 2021, S. 212
[35] vgl. Mohri et al., 2018, S. 7
[36] vgl. Bishop und Nasrabadi, 2006, S. 3

Da sich diese Arbeit ausschließlich mit der Lernmethode des überwachten Lernens beschäftig, wird im nächsten Unterkapitel genauer auf die beiden Lerntechniken des Supervised Machine Learnings eingegangen, der *Regression* sowie der *Klassifikation*.

2.2.4 Aufgabentypen des SML

2.2.4.1 Regression

Wie bereits im vorangehenden Kapitel erwähnt, stellt die Regression eine der wichtigsten Aufgabentypen der überwachten Lernmethode dar. In der Statistik beschreibt eine Regressionsanalyse ein *„Werkzeug zum Aufbau mathematischer und statistischer Modelle, die Beziehungen zwischen einer abhängigen Variablen und einer oder mehreren unabhängigen oder erklärenden Variablen charakterisieren, die alle numerisch sind.“* [37]

Liegt ein Satz von numerischen Datenpunkten vor, der sowohl eine Eingabe x als auch eine Ausgabe y enthält, so besteht die Aufgabe der Regression darin, eine so genannte Regressionsfunktion abzuleiten. Diese Funktion soll eine verborgene funktionale Beziehung von x und y erkennen und abbilden. Die Funktion kann etwa einer einfachen linearen Annäherung oder einer Polynomfunktion höheren Grades entsprechen. [38]

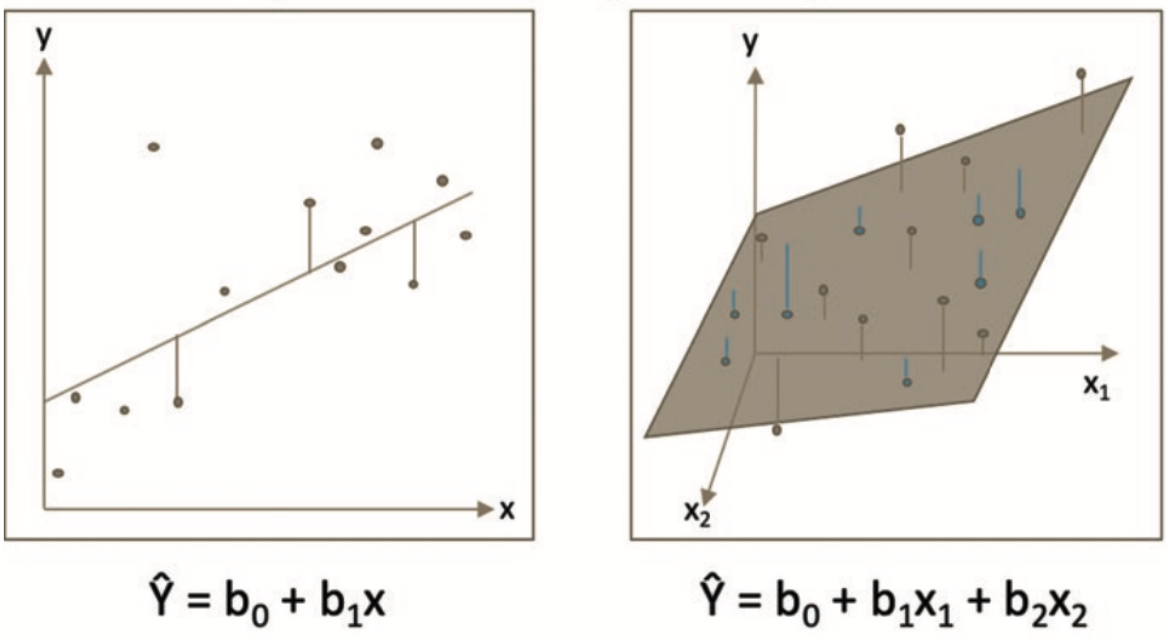

Abbildung 4: Einfache (links) und Multiple Regressionsanalyse (rechts)[39]

[37] Weber, 2020, S. 57
[38] vgl. Sammut und Webb, 2011, S. 838
[39] Weber, 2020, S. 59

Im Machine Learning spricht man von Regressionsaufgaben, wenn das Hauptziel der Aufgabe der Vorhersage von numerischen Zielwerten entspricht. Bei regressionsbasierten Methoden wird das Modell zunächst mit Eingabe- und Ausgabedatenproben antrainiert und dadurch die Beziehungen und Assoziationen angelernt. Mit diesem Wissen soll das Modell für unbekannte Datensätze kontinuierliche und numerische Ausgabewerte liefern. Für die Vorhersage von numerischen Zielwerten, stehen in den überwachten Lernmethoden eine Reihe von Regressionsmodellen zur Verfügung.[40]

In Abbildung 4 sind die für diese Arbeit wesentlichen Typen von Regressionsmodellen dargestellt. Ein Regressionsmodell, das eine einzelne unabhängige Eingangsvariable x und eine Zielvariable y beinhaltet, wird als *lineare* oder *einfache Regressio*n bezeichnet. Vergrößert sich die Anzahl der unabhängigen Eingangsvariablen x, so spricht man von einem *multiplen Regressionsmodell.* Bei der multiplen Regression werden anhand mehrerer Merkmale die Zielgröße vorhergesagt. Neben den genannten Regressionsmodellen gibt es noch weitere Formen der Regression, wie etwa die *polynomiale Regression*, die *nichtlineare Regression* oder die *logistische Regression.* Letztere wird etwa für kategoriale Daten verwendet wird.[41]

2.2.4.2 Klassifikation

Bei der Klassifikation handelt es sich, neben der Regression, um den zweiten wichtigen Aufgabentypen für das überwachte maschinelle Lernen. Das Wort Klassifikation wird im allgemeinen Sprachgebrauch dafür verwendet, Daten in Kategorien einzuteilen, die sich sinnvoll zusammenfassen lassen. Im Machine Learning wird der Begriff der Klassifikation für eine Art des Lernens verwendet, in der ein Lernalgorithmus eine Eingangsvariable einer Klasse zuordnet. Dafür werden dem Modell, wie es beim Supervised Learning üblich ist, zuvor Beispieldaten der Klassen mitsamt ihren Eigenschaften übergeben. Der Algorithmus generiert einen *Klassifikator* anhand der Beispieldaten und kann damit neue Beispiele, deren Klasse nicht bekannt ist, den zuvor erstellten Klassen zuordnen.[42]

Man unterscheidet im Allgemeinen zwischen der *binären Klassifikation* und einem *Mehrklassenproblem*. Bei der binären Klassifikation werden Objekte auf zwei Klassen, wie etwa wahr oder falsch, positiv oder negativ, abgebildet. Als Beispiel könnte eine eingegangene E-Mail betrachtet werden, die entweder einen Spam enthält oder nicht. Im Unterschied dazu, können bei einem Mehrklassenproblem die Objekte auch mehrere Klassen oder gar keinen Klassen zugeordnet werden.[43]

[40] vgl. Sarkar et al., 2018, S. 37
[41] vgl. Sarkar et al., 2018, S. 38
[42] vgl. Sammut und Webb, 2011, S. 168
[43] vgl. Schacht & Lanquillon, 2019, S. 103 f

2.2.5 Machine Learning Algorithmen

Um Datenprobleme mithilfe des maschinellen Lernens lösen zu können, werden in der Literatur viele verschiedene Algorithmen beschrieben, die laufend ergänz werden. Neue Ansätze weisen jedoch oft nur geringe Unterschiede zu den bekannten Algorithmen auf. Aufgrund der hohen Anzahl unterschiedlicher Algorithmen, die im Machine Learning gebrauch finden, ist ein vollständiger Überblick aller Algorithmen nicht mehr möglich.[44] Zudem weisen Datenwissenschaftler gerne darauf hin, dass es keinen „Allrounder-Algorithmus“ gibt, der für die Lösung eines Problems am besten geeignet ist. Die Auswahl des verwendeten Algorithmus hängt von der Art des Problems ab.[45]

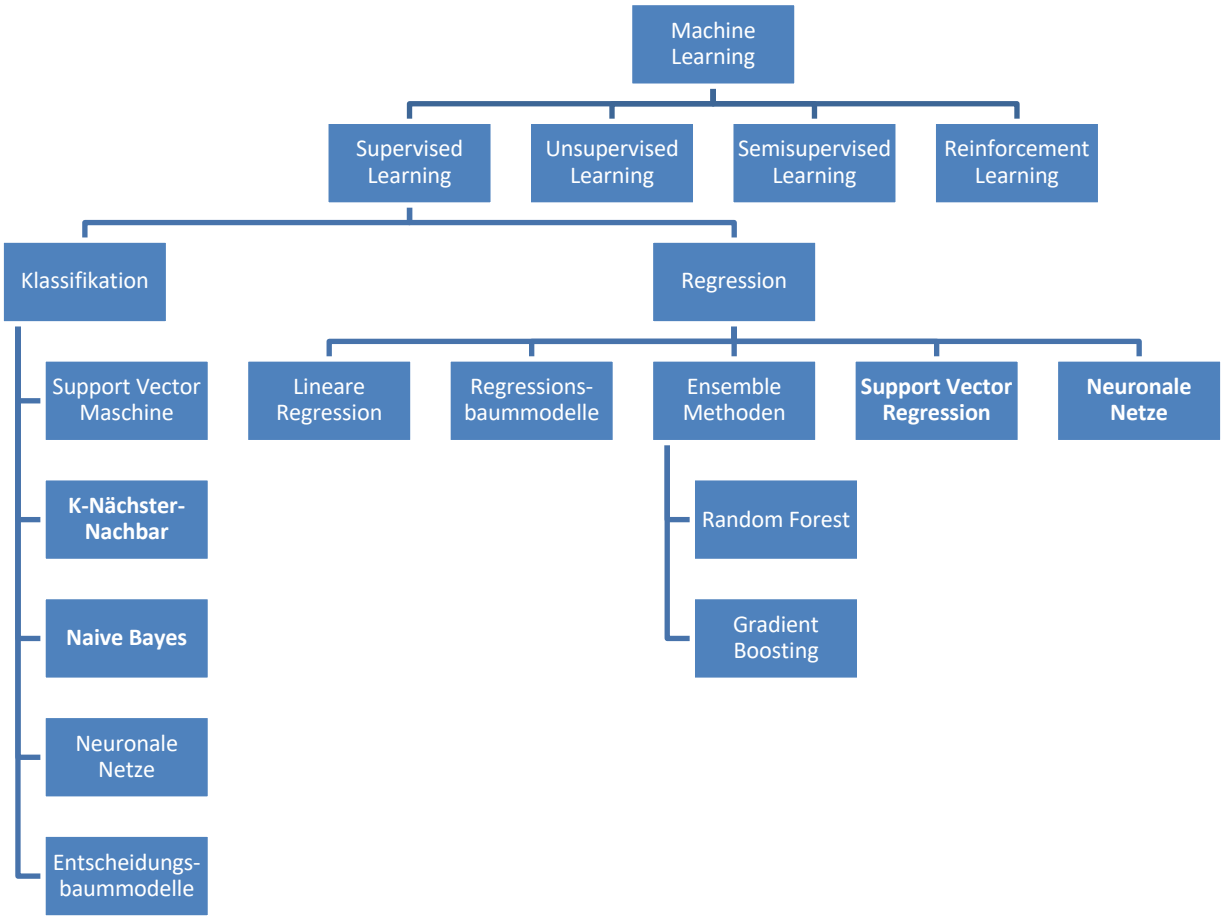

Abbildung 5: Übersicht über wesentliche SML-Algorithmen[46]

[44] vgl. Schacht & Lanquillon, 2019, S. 122
[45] vgl. Mahesh, 2020, S. 1
[46] Eigene Darstellung; vgl. Sharma und Kumar, 2017, S. 1551; vgl. Mahesh, 2020; vgl. Schacht und Lanquillon, 2019, S. 126ff

Abbildung 5 zeigt eine Übersicht über die gängigen Machine Learning Algorithmen aus der überwachten Lernmethode, die in der Literatur häufigen Gebrauch finden. Eine genaue Beschreibung der für diese Arbeit wesentlichen SML-Algorithmen wird im Rahmen der Modellierung in Kapitel 4.6 vorgenommen.

2.3 Feature Engineering

2.3.1 Features und ihre Daten

Beim maschinellen Lernen, Data Mining und der Datenanalyse ist ein **Feature** ein Attribut oder eine Variable, welches zur Beschreibung von Datenobjekten verwendet wird. Die Begriffe *Feature, Merkmal, Attribut* oder *Variable* werden häufig als Synonym verwendet. Als Beispiel für ein Feature könnte das Alter oder die Augenfarbe einer Person genannt werden. Features bilden die Grundlage für die Datenanalyse und sind für die Erstellung von Machine Learning Modellen essenziell. Jedes Objekt nimmt für jedes Feature einen bestimmten Wert an, wodurch sich eine zweidimensionale vektorbasierte Darstellung der Datenobjekte ergibt. [47] Beim SML existiert daneben noch die Zielgröße, wie sie in Abbildung 6 zu sehen ist.

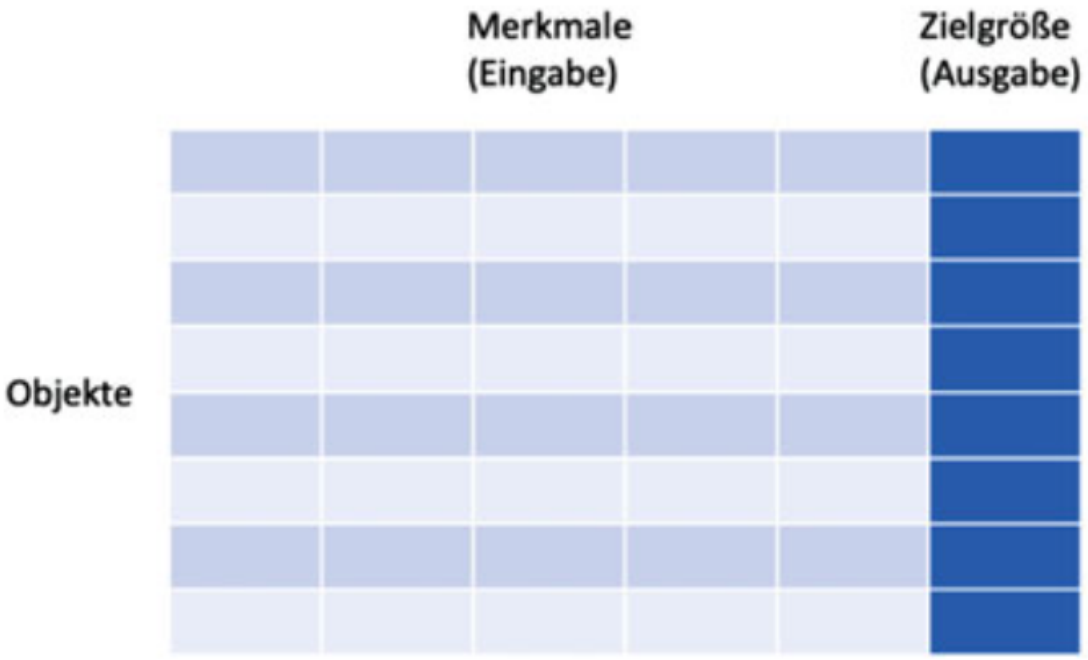

Abbildung 6: Matrixdarstellung der Features im SML[48]

[47] vgl. Dong und Liu, 2018, S. 2
[48] Schacht & Lanquillon, 2019, S. 102

In der höchsten Unterteilungsebene können Daten in *quantitative* und *qualitative Daten* unterteilt werden. Quantitative Daten sind Daten, die numerischer Natur sind. Qualitative Daten sind Daten, die kategorialer Natur sind und die Qualität eines Objekts beschreiben.[49] Die Features werden ebenfalls in verschiedene Merkmalstypen unterteilt und benötigen aufgrund struktureller Unterschiede in ihrem Bereich auch unterschiedliche Arten der Analyse. So können die Features unter anderem kategorisch, ordinal oder numerischer Natur sein. Kategorische Features beinhalten diskrete Werte, wie etwa Farben oder Marken. Im einfachsten Fall kann es sich dabei um binäre Werte handeln. Ordinale Features stellen eine Menge von geordneten Werten dar, wie etwa den Grad eines Titels. Mithilfe von numerischen Features werden kontinuierliche Werte abgebildet. Für Machine Learning Algorithmen müssen kategorische Features meist in numerische Features umgewandelt werden. Somit können auch numerische Features herangezogen werden, um diskrete Variablen darzustellen.[50] Eine Gegenüberstellung und Auflistung der Datentypen wurde zur besseren Veranschaulichung in Tabelle 1 durchgeführt.

Datentyp	Merkmalstyp	Werte	Beispiel
qualitativ	ordinal	geordnet	{Bachelor, Master, Doktor}
	kategorisch	diskret	{blau, schwarz, grün}
quantitativ	numerisch	kontinuierlich, diskret	{1.43, 178.3, 9.45} {1, 2, 3}

Tabelle 1: Unterschiedliche Feature-Typen und ihre enthaltenen Werte[51]

2.3.2 Definition: Feature Engineering

Die Eingabe für ein SML-System erfolgt mit einer Reihe von Trainingsbeispielen, die im Vorfeld zur Verfügung stehen müssen. Diese Trainingsdaten sind in der Regel Rohdaten, die als Ergebnis einer Datenerhebung erfasst werden. Der Datensatz besteht aus einem Eingangsvektor (Feature-Vektor), der wiederum aus mehreren Features zusammengesetzt ist, sowie dem Ausgangsvektor, der die Zielwerte beinhaltet. Je nachdem, ob die Zielgröße eine diskrete Größe (Klassifizierung) oder eine kontinuierliche Größe (Regression) darstellt, handelt es sich um ein Klassifizierungs- oder Regressionsproblem, das vorab definiert wird.[52]

In der PPS handelt es sich meist um kontinuierliche Werte bei der Vorhersage von Zielgrößen. Der auf dieser Arbeit zugrundeliegende Datensatz beinhaltet ebenfalls kontinuierliche Größen als Zielgrößen. Daher wird in diesem Kapitel das Feature Engineering, bezogen auf Regressionsprobleme, behandelt.

[49] vgl. Ozdemir und Susarla, 2018, S. 54
[50] vgl. Dong und Liu, 2018, S. 2
[51] Eigene Darstellung
[52] vgl. Duboue, 2020, S. 6

Nachdem die Rohdaten aus der Datenerhebung erfasst wurden, müssen die Datensätze in der Regel zuvor für das jeweilige maschinelle Lernmodell vorbereitet werden. An dieser Stelle findet das Feature Engineering seinen Einsatz und stellt einen entschiedenen Schritt für die Erstellung eines hochwertigen Machine Learning Modells dar. Im Rahmen des Feature Engineering werden die Features durch gewissen Techniken aus den Rohdaten extrahiert und in Formate umgewandelt, die für das Lernmodell geeignet sind. Dieser Prozess wird in Abbildung 7 veranschaulicht.[53]

Im Wesentlichen sorgt das Feature Engineering dadurch für eine qualitativ bessere Leistung des Modells, sowie einer Minimierung der Berechnungszeiten. In der Literatur wird das Verfahren des Feature Engineering als den aufwendigsten Schritt bei der Erstellung eines Machine Learning Modells beschrieben. Trotz seiner Bedeutung findet die Technik jedoch selten ihren Einsatz, da eine Verallgemeinerung einer Herangehensweise, aufgrund der großen Unterschiede in der Datenstruktur und den Modellen, nur schwierig zu erstellen ist. [54]

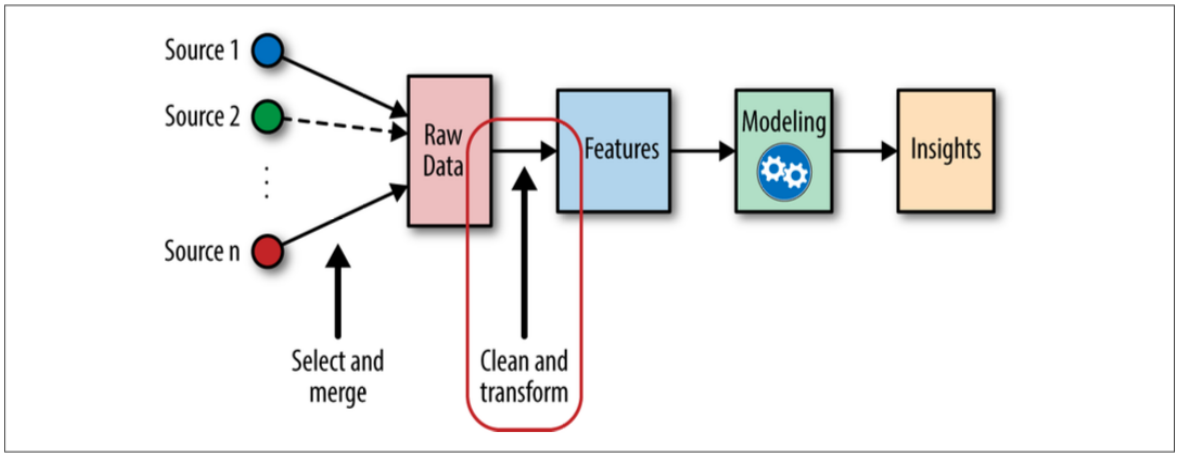

Abbildung 7: Einsatz von Feature Engineering in der Machine Learning Pipeline[55]

Die Entwicklung und Auswahl von Features ist kein einmaliger Prozess, der ad hoc durchgeführt werden kann, sondern findet (etwa nach dem CRISP-DM Prinzip) iterativ statt. So muss die Auswahl der Features gegeben falls für jedes Modell neu stattfinden und getestet werden. Dies unterstreicht den großen Aufwand, der hinter dem Feature Engineering stehen kann. Der Arbeitsablauf im Feature Engineering erfolgt nach Sarkar et al. in drei Hauptschritten, dem *Feature Extraction and Engineering, Feature*

[53] vgl. Zheng und Casari, 2018, S. vii
[54] vgl. Zheng und Casari, 2018, S. vii
[55] Zheng und Casari, 2018, S. 4

Scaling und dem Feature Selection. Diese Arbeitsschritte werden im weiteren Verlauf genauer erläutert.[56]

2.3.3 Arbeitsablauf im Feature Engineering

2.3.3.1 Feature Extraction and Engineering

Die Begriffe Feature Extraction und Feature Engineering werden in der Regel als Synonyme verwendet und beschreiben einen Prozess zur Umwandlung von Rohdaten in Features. Der Arbeitsschritt Feature Extraction and Engineering umfasst etwa den Umgang mit fehlerhaften Daten, das Einfügen fehlender Werte oder die Umwandlung bestimmter Werte. Dieser Prozess wird in der Pipeline zur Entwicklung eines Machine Learning Modells als einer der wichtigsten Prozesse beschrieben. Dabei gelten für das Feature Extraction keine festen Regeln. Um die Rohdaten in geeignete Features umzuwandeln, benötigt es eine Kombination aus Fachwissen, ein Verständnis für jeweiligen Prozesse, sowie mathematische, aber auch von Hand getätigte Transformationen. Die Art des Datentyps (siehe 2.3.1) ist schlussendlich dafür verantwortlich welche Technik des Feature Extractions angewendet werden kann.[57]

2.3.3.2 Feature Scaling

Einige Machine Learning Algorithmen, wie die lineare Regression, reagieren empfindlich auf die Größe oder den Umfang von Werten. Für diese Modelle kann es sinnvoll sein eine Skalierung der Features vorzunehmen. Beim Feature Scaling werden die Daten so transformiert, dass sie am Ende innerhalb eines bestimmten Intervalls liegen. Einige Skalierungsmethoden wie Normalisierung und Standardisierung verändern auch die Verteilung der Daten, welche für gewisse Modelle, wie etwa Neuronale Netze, eine Verbesserung der Performance mit sich bringen können.[58]

2.3.3.3 Feature Selection

Je mehr Features ein Datensatz enthält, desto komplexer und schwieriger wird es die Modelle zu interpretieren. So besteht das Ziel der Feature Selection darin, eine optimale Anzahl an Features für das Training des Modells auszuwählen, um so die Berechnungszeit zu verkürzen und die Komplexität des Modells zu verringern. Die Strategien zur Feature Selection unterteilt man dabei in drei Hauptkategorien - *Filter Methods, Wrapper Methods* und *Embedded Methods:*[59]

Bei den **Filter Methoden** werden jene Features bereits im Vorfeld aussortiert, die für das Modell mit sehr hoher Wahrscheinlichkeit nicht nützlich sein werden. Dazu werden

[56] vgl. Sarkar et al., 2018, S. 177
[57] vgl. Sarkar et al., 2018, S. 181 ff
[58] vgl. Cellier und Driessens, 2020, S. 10
[59] vgl. Sarkar et al., 2018, S. 242

zum Beispiel Korrelationen zwischen den Features ermittelt und mithilfe eines Schwellenwertes aussortiert. Dabei gilt es darauf zu achten keine nützlichen Features versehentlich zu eliminieren. Die Filter Technik ist eine kostengünstige Technik, die jedoch nicht die Art des verwendeten Modells berücksichtigt. Unterschiedliche Features können für unterschiedliche Modelle andere Ergebnisse erzielen.[60]

Die **Wrapper Methoden** ermöglichen es, die Interaktion mehrere Merkmale zu erfassen und die beste Teilmenge an Features auszuwählen. Dafür werden in einer eigenen Methode Teilmengen von Features anhand des Modells getestet und hinsichtlich ihrer Qualität untersucht und iterativ verfeinert. Im Vergleich zur Filter Methode ist diese Methode kostenintensiv und aufwendig, nimmt jedoch Rücksicht auf das ausgewählte Modell.[61]

Als **Embedded Methods**, zu deutsch *eingebettete Methoden*, werden jene Methoden des Feature Selections bezeichnet, bei denen die Auswahl der Features als Teilarbeit der Machine Learning Algorithmen selbst durchgeführt wird. Die Auswahl der Features erfolgt dabei nicht durch das Entfernen der unbrauchbaren Features aus den Trainingsdaten, sondern wird durch Selektion des Modells erstellt. Als Beispiel für eine eingebette Methode ist der Entscheidungsbaum (Decision Tree) zu nennen.[62]

[60] vgl. Zheng und Casari, 2018, S. 38
[61] vgl. Zheng und Casari, 2018, S. 38
[62] vgl. Duboue, 2020, S. 94f

3 Stand der Technik

In diesem Kapitel wird ein Überblick über den aktuellen Stand der Technik hinsichtlich der Anwendung von maschinellem Lernen in der PPS gegeben. Besonderes Augenmerk richtet sich dabei auf die Anwendung von Algorithmen des SML, die in der PPS zur Ermittlung von Zielgrößen zum Einsatz kommen. Dazu werden zunächst wesentliche Studien zur Anwendung von SML in der PPS vorgestellt und diese anschließend in einer Tabelle in der Zusammenfassung einander gegenübergestellt. Daraus sollen bedeutende SML-Algorithmen aus der Literatur identifiziert werden und der aktuelle Stand der Technik aufgezeigt werden. Weiters werden erste Ansätze zur Definition und Implementierung der Planungsqualität in der Produktionsplanung vorgestellt.

3.1 Methodisches Vorgehen

Zur Erhebung des aktuellen Forschungsstand wurde eine Literaturanalyse durchgeführt. Dazu wurden zunächst Stichwörter ausgewählt, welche anschließend in geeigneten Online- Datenbanken zur Suche verwendet wurden. Dabei wurde ein definierter Zeitraum der Veröffentlichung festgelegt. Vorzugsweise wurden wissenschaftliche Publikationen herangezogen. Anhand des Titels, der Schlagwörter und des Lesens der Einleitung wurde anschließend eine Vorauswahl der Literatur getroffen. Bei dieser Vorauswahl werden 35 Studien herangezogen, die genauer untersucht werden. Schließlich wurden Kriterien zur Auswahl der geeigneten Literatur festgelegt. Nach der vollständigen Durchsicht der vorausgewählten Literatur wird die finale Auswahl der Literatur anhand der vordefinierten Kriterien durchgeführt.[63] Abbildung 8 zeigt die Schritt- für- Schritt Anleitung für die Vorgehensweise der Literaturauswahl für Kapitel 3.

Schritt	**Beschreibung**	
1	Stichwörter-Formulierung	<u>Verwendete Stichwörter:</u> (Supervised) Machine Learning, Algorithms, Production Planning and Control, Production Planning and Scheduling, Planning, Feauture Selection, Feature Engineering, Planning Quality, Robust Planning, Definition, Stability,

[63] vgl. Kirchner und Meyer, 2022, S. 132ff

2	Auswahl der Datenbanken	Verwendete Datenbanken: • Google Scholar • Science Direct • IEEE Xplore • Springer
3	Zeitraum definieren	Zeitraum: 2015-2023
4	Vorauswahl der Literatur	Filter: Titel -> Schlagwörter -> Einleitung
5	Kriterien definieren	Kriterien zur Literaturauswahl: Kapitel 3.2: • Mehrere SML- Algorithmen wurden verwendet und untersucht. • Es wurde eine Evaluierung der SML-Algorithmen durchgeführt. • Es wurden Machine Learning Algorithmen aus der überwachten Lernmethode herangezogen. • Die Anwendung der SML- Algorithmen erfolgte in der Produktion/ Fertigung. • Die SML- Algorithmen unterstützen die Produktionsplanung. Kapitel 3.3: • Definition der Planungsqualität Kapitel 3.4: • Wissenschaftlicher Ansatz zur Unterstützung der PPS durch ML- Algorithmen
6	Finale Auswahl	Anhand der in 5 definierten Kriterien

Abbildung 8: Vorgehensweise zur Literaturanalyse[64]

[64] Eigene Darstellung

3.2 Ähnliche Lösungsansätze zur Anwendung von SML in der PPS unter Zuhilfenahme von Feature Engineering

Pfeiffer et al., 2016:

Pfeiffer et al. verglichen die Ergebnisse von drei Machine Learning Modellen zur Vorhersage der Durchlaufzeit in der Fertigung durch die Kombination von Simulation und statistischen Lernmethoden (multivariater Regression). Die Modelle wurden mit zuvor erhobenen Protokolldaten trainiert, validiert und getestet.

Mithilfe von Feature Selection wurden die relevanten Features aus einer großen Anzahl von Variablen ausgewählt. Um die geeigneten Features auszuwählen, haben Pfeiffer et al. zunächst Annahmen über die einzelnen Merkmale getroffen, welche sie dann in einer *Korrelationsmatrix* überprüft haben. Weiters wurde eine Explorative Analyse der Daten hinsichtlich der Datenstruktur vorgenommen. In der Studie werden Regressionsbäume (RT), eine multiple lineare Regression (MLR) und ein Random Forrest (RF) Ansatz durchgeführt. Das von ihnen entwickelte Random-Forest-Modell übertraf die Genauigkeit der linearen Regression und der Regressionsbaummodelle.[65]

Lingitz et al., 2018:

Lingitz et al. testen in ihrer Studie Machine Learning-Algorithmen, um die *Vorlaufzeit* (engl.: *Lead Time*) eines Halbleiterunternehmens vorherzusagen. Sie untersuchen dafür den Einsatz von Regressionsalgorithmen und ihre Auswirkungen auf die Erhöhung der Genauigkeit der Vorlaufzeit-Vorhersage. Sie führen einen Multikriterien-Vergleich durch und schließen dadurch Schlussfolgerungen für die Auswahl von Features und die Anwendbarkeit der Machine Learning Methoden in der Halbleiterindustrie.

Bei ihrer Analyse wurden elf verschiedene Lernmethoden aus dem Machine Learning unter der Anwendung von R und R Studio bewertet. Zunächst testen sie drei lineare Modelle, ein lineares Regressionsmodell (LM), sowie ein Ridge und ein Lasso Modell, die in etwa die gleiche Vorhersagegenauigkeit erzielen. Weiters testen sie drei unterschiedliche Arten von Regressionsbäumen. Sie testen ein Support Vector Regressionsmodell (SVM), Multivariate adaptive Regressionssplines (MARS), eine Nächste-Nachbarn-Klassifikation (kNN), sowie einen Neuronalen Netzwerk Ansatz (ANN).

[65] vgl. Pfeiffer et al., 2016

	LM	Ridge	Lasso	RT	bagged RT	RF	boosted RT	SVM	MARS	kNN	ANN
MAE	487	510	508	563	394	390	397	423	488	504	535
MAPE	42.7	45.0	44.7	53.5	33.9	33.8	33.9	30.9	43.15	44.0	53.4
MSE	529408	573520	572939	639617	369993	360780	369414	500693	513638	554897	658852
RMSE	727	757	756	799	608	600	607	707	716	745	771
NRMSE	15.2	15.8	15.8	16.7	12.7	12.5	12.7	14.77	14.9	15.5	17.3

Abbildung 9: Ergebnisse der Analyse von Lingitz et al.

Während der Modellerstellung und Auswahl der Features wurde eine 10 – Fache Kreuzvalidierung durchgeführt. Die Genauigkeit der Modelle haben Lingitz et al. mit 5 verschiedenen Fehlermaßstäben gemessen, nämlich mit dem mittleren absoluten Fehler (MAE), dem mittleren absoluten prozentualen Fehler (MAPE), dem mittleren quadratischen Fehler (MSE), dem mittleren quadratischen Fehler (RMSE) und dem normalisierten mittleren quadratischen Fehler (NRMSE), wobei der NRMSE den durchschnittlichen Vorhersagefehler in Prozent des tatsächlichen Wertes angibt. Eine genauere Beschreibung der Fehlerfunktionen findet sich in Kapitel 4.7.1.

Der Einfluss der Features auf die Modelle wurde mithilfe einer Sensitivitätsanalyse untersucht. So konnten jene Variablen ermittelt werden, die einen negativen Einfluss auf die Modelle haben. Das beste Ergebnis liefert in ihrer Studie der Random Forest Ansatz (RF) mithilfe der Sensitivitätsanalyse und der Merkmalsauswahl. Die zusammenfassenden Ergebnisse sind in Abbildung 9 zu sehen.[66]

Gyulai et al., 2018:

Gyulai et al. analysieren und vergleichen analytische und maschinelle Vorhersagemodelle zur Prognose der Durchlaufzeit in der Fließbandfertigung für optische Brillengläser. Die maschinellen Lernmethoden, die sie in ihrer Studie heranziehen, sind die lineare Regression, Regressionsbäume, ein Random Forest Modell und die Support-Vektor-Regression. In allen Fällen wurden die Auswahl der Features und die Feinabstimmung der Parameter mit einer *schrittweisen Vorwärtsauswahl* durchgeführt. Die Wichtigkeitseinstufung wurde durch ein Random Forest Modell unterstützt.

Aus ihren Testergebnisse geht hervor, dass die analytischen Prognosemodelle bei der Vorhersage der Durchlaufzeit wesentlich schlechter abschneiden als jene Prognosemodelle, die auf maschinellem Lernen beruhen. Die Autoren betonen, dass die Auswahl des geeigneten Algorithmus von den vorherzusagenden Parametern und dem untersuchten Prozess abhängt. In ihrem Fall liefern alle Modelle zufriedenstellende Ergebnisse. Das beste Ergebnis liefert der Random Forest Ansatz.[67]

[66] vgl. Lingitz et al., 2018
[67] vgl. Gyulai et al., 2018

Bender und Ovtcharova, 2021:

Bender und Ovtcharova untersuchen einen Ansatz zur Ermittlung der Durchlaufzeiten im Bereich der Auftrags- und Kleinserienfertigung, indem sie ein Auto Machine Learning (AutoML) Modell anwenden. Für die Vorhersage verwenden sie sowohl Auftragsdaten aus einem ERP-System als Echtzeitdaten aus der Fabrik, die von einer IoT-Plattform übermittelt werden. AutoML-Systeme ermöglichen ein schnelles Training und Benchmarking von Modellen, jedoch auf Kosten der Modellqualität.

Bei den Trainingsdaten handelt es sich um aus einer Simulation erzeugte und exportierte Daten, die 32 Features enthalten. Zur Aufbereitung der Daten verwenden die Autoren einen Label-Encoder. Zur Erstellung der Modelle wird eine von H20.ai[68] verwendete AutoML-Funktion verwendet und das Beste Modell auf Grundlage des RMSE ausgewählt. Das beste Gesamtergebnis liefert die Gradient Boosting Machines (GBM). Die Geschwindigkeit und Einfachheit der Bedienung eines AutoML-Systems, im Vergleich zu konventionellen ML-Methoden, ist ein signifikanter Vorteil, auch wenn dies mit einer geringeren Qualität der Modelle einhergeht.[69]

Gahm et al.,2022:

In der *hierarchischen Produktionsplanung* ist es unerlässlich Interdependenzen zwischen Entscheidungen der obersten Instanz und den untergeordneten Managementebenen zu berücksichtigen. In ihrer Studie nutzen Gahm et al. Regressionsmodelle aus dem Machine Learning zur Vorhersage der erwarteten Reaktionen der unteren Manager Ebene auf Entscheidungen des Top Managements. Dafür nutzen sie Daten eines Unternehmens aus der metallverarbeitenden Industrie, genauer aus der blechverarbeitenden Industrie. Um den Datensatz vorzubereiten und unnützliche Features auszusortieren, nutzen die Autoren eine Methode zur Dimensionsreduktion, die Hauptkomponentenanalyse (PCA). Außerdem werden weitere Methoden zur Dimensionsreduktion, wie die "Kernel PCA", "Locally Linear Embedding" oder "Linear and Quadratic Discriminant Analysis“, erwähnt. Zur Auswahl der Features wird schließlich *Hyper Parameter Tuning* angwendet.

Die Autoren testen eine sehr große Anzahl von 18 Machine Learning Algorithmen: Mehrfache lineare Regression (MLR), Ridge-Regression (RR), Lasso-Regression (LR), Elastisches Netz (EN), LARS-Lasso (LL), Polynomiale Regression (PR), Stochastic Gradient Descent (SGD), K-nearest Neighbors Regression (KNN), Kernel Ridge Regression (KRR), Support Vector Regression (SVR), Random Forest Regressor, Neuronales Netzwerk, sowie mehrerer Arten von Regressionsbäumen. Als Kennzahl zur Evaluierung der Testergebnisse nutzen sie den RMSE. Alle Ergebnisse

[68] https://www.h2o.ai/, (letzter Zugriff: 24.01.2023)
[69] vgl. Bender und Ovtcharova, 2021

zeigen eine sehr gute Prognosegenauigkeit der Zielgrößen. Das beste Ergebnis erzielt ihr Modell eines künstlich neuronalen Netzwerkes.[70]

3.3 Planungsqualität in der PPS

Um die Prognosegenauigkeit für Zielgrößen in der PPS zu verbessern und die Zuverlässigkeit eines Modells beurteilen zu können, muss die *Planungsqualität* des Modells herangezogen werden. In der Literatur gibt es bisher keine einheitliche mathematische Definition und kein einheitliches Verständnis des Begriffes der Planungsqualität im Zusammenhang mit der PPS. In ihrer Studie versuchen Rayback et al. den Begriff der Planungsqualität für die PPS zu definieren und stellen zwei Ansätze vor, die zur Verringerung der Abweichungen der prognostizierten Werte und den tatsächlichen Werten verwendet werden können. Die Autoren definieren eine hohe Planungsqualität wie folgt: *„Ideally no deviation, but a deviation in an at least an acceptable range (which can vary depending on different industrial context), between predicted times and actual times (e.g., lead time, set up time, operation time) exists."* Im Idealfall gibt es also keine Abweichungen zwischen den tatsächlichen Zeiten und den vorhergesagten Zeiten.[71]

Lucht et al. definieren die Planungsqualität etwas detaillierter. Sie unterteilen die Planungsqualität in die *Planungsgenauigkeit* und die *Planstabilität*. Die Planungsgenauigkeit beschreibt den Grad der Übereinstimmung eines geplanten Ereignisses mit seiner tatsächlichen Realisierung. Die Planstabilität dient als Maß für die Konsistenz sowie das frühzeitige Erkennen möglicher Änderungen einer Planung. Mit anderen Worten sorgt eine hohe Planungsstabilität für keine oder nur geringe Anpassungen durch eine Planungsiteration. Zusammen ergeben die Planungsgenauigkeit und die Planstabilität die Planungsqualität. Diese kann nach Ansicht der Autoren sowohl im zeitlichen Verlauf des Auftragsabwicklungsprozesses als auch zu einem einzelnen Zeitpunkt betrachtet werden. Im zeitlichen Verlauf können mehrere Fertigungsaufträge hinsichtlich ihrer Planungsqualität untersucht und zu einem definierten Zeitpunkt anschließend miteinander verglichen werden. Somit ergeben sich vier Perspektiven, die in Abbildung 10 dargestellt sind.[72]

[70] vgl. Gahm et al., 2022
[71] vgl. Ryback et al., 2019, S. 132
[72] vgl. Lucht et al., 2021

	Planungsqualität	
	Planstabilität	**Planungsgenauigkeit**
einzelner Zeitpunkt	Ausmaß der Veränderung der geplanten Termine/Inhalte eines Beobachtungsobjekts zu einem definierten Zeitpunkt entlang seiner Prozesskette	Genauigkeit bzw. Grad der Übereinstimmung des Planungsergebnisses zu einem definierten Zeitpunkt entlang der Prozesskette mit dem tatsächlich realisierten Ergebnis
zeitlicher Verlauf	Anzahl, Abstand und Intensität von Veränderungen eines geplanten Datums / Inhalts des Betrachtungsgegenstandes entlang der Prozesskette des Betrachtungsgegenstandes	Zeitliche Entwicklung / Veränderung der Planungsgenauigkeit bzw. -abweichung entlang der Prozesskette im Verhältnis zum tatsächlich realisierten Ergebnis.

Abbildung 10: Perspektiven der Planungsqualität nach Lucht et al.

Die Frage nach der Qualität eines Produktionsplans stellen sich auch Kempf et al. Sie untersuchen, was einen „guten" Produktionsplan ausmacht. Dazu wird zunächst die Machbarkeit sowie die Akzeptanz eines Planes überprüft. Die *Machbarkeit* ist dann gewährleistet, wenn keine der zuvor definierten Randbedingungen verletzt wird. Der Plan muss also physisch umsetzbar sein. Die *Akzeptanz* eines Planes bedeutet hingegen, dass dieser nicht durch triviale Änderungen verbessert werden kann. Zur Definition und Bewertung der Planungsqualität ziehen die Autoren mehrere Perspektiven heran. So spielt zum einen die Anzahl der zu betrachtenden Produktionspläne eine Rolle, wobei die Komplexität der Qualitätsbewertung mit steigender Anzahl zunimmt. Zum anderen muss für die Messung der Planungsqualität zwischen der absoluten und der relativen Bewertung unterschieden werden. Als Maßstab zur absoluten Bewertung kann etwa die beste Leistung unter perfekten Bedingungen herangezogen werden, die in der Produktion erzielt wurde. Bei der relativen Bewertung werden mehrere Produktionspläne miteinander verglichen und der Beste herangezogen. [73]

Im Zusammenhang mit der Planungsqualität fällt immer wieder der Begriff des **Robust Planning.** Der Begriff wird in der Literatur bisher viel diskutiert und verwendet, führt aber hinsichtlich der Definition der Planungsqualität zu Missverständnissen.

[73] vgl. Kempf et al., 2000

Ein Produktionsplan gilt als robust, wenn er selbst dann zu einem akzeptablen Ergebnis führt, wenn während der Ausführung des Plans unvorhersehbare Ereignisse auftreten. Dafür benötige es die Anwendung mathematischer Modelle, die Unsicherheiten, wie Maschinenausfälle, in der Produktion berücksichtigen können. Zur Messung der Robustheit werden verschiedene KPIs verwendet, wobei in den meisten Fällen die Differenz zwischen dem geplanten und dem tatsächlichen Output der Produktion betrachtet wird.[74]

3.4 Ansatz zur Implementierung der Planungsqualität in der PPS

Um die Planungsqualität zu erhöhen, entwickeln Ryback et al. zwei Ansätze, die auf der Anwendung von Machine Learning Algorithmen beruhen:

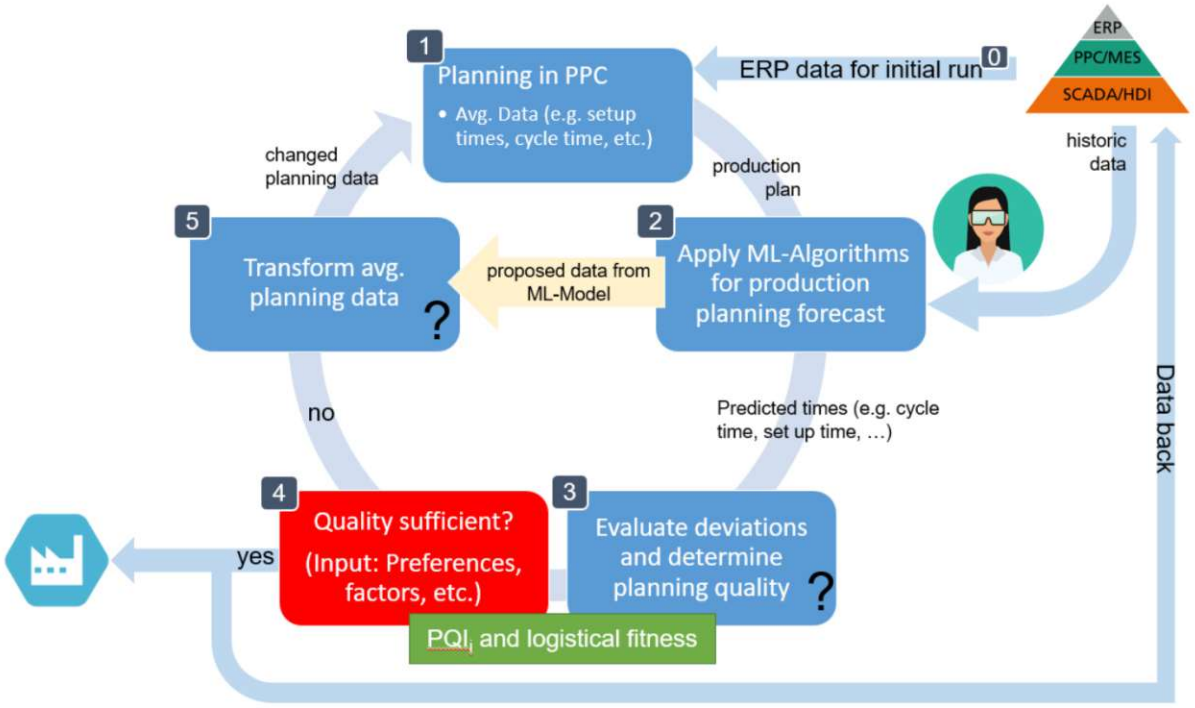

Abbildung 11: Evolutionary Approach nach Ryback et al.

Der Evolutionary Approach ist als zusätzliches Entscheidungsunterstützungssystem zum eigentlichen Planungssystem gedacht. Der Ansatz besteht aus sechst Schritten, wobei die Schritte 1 bis 5 in iterativen Schleifen ablaufen. Ähnlich wie das Prinzip der *kontinuierlichen Verbesserung*, soll die Planungsqualität mit jeder Iteration gesteigert werden. Die Planung wird am Beginn (Schritt 1) durch ein konventionelles Planungssystem, etwa durch ein ERP-System, durchgeführt. Dieser Produktionsplan dient als Eintritt in den Zyklus. Im nächsten Schritt werden unter der Zuhilfenahme von

[74] vgl. Gyulai et al., 2017

Machine Learning Prognosezeiten vorhergesagt. Innerhalb des Zyklus wird ein Vergleich der geplanten und der prognostizierten Zeiten der SML- Algorithmen durchgeführt und die Planungsqualität in Form geeigneter Kennzahlen (*Key Performance Indicators - KPIs*) und einem eigens eingeführten Planungsqualitätsindex (PQI) an den Planer rückgemeldet, wie es in Abbildung 11 dargestellt ist. Mit dieser Methode sollen auch Faktoren ermittelt werden, die eine schlechte Planungsqualität begünstigen. Sollte der Planer mit dem Ergebnis nicht zufrieden sein, wird der Prozess nach einer Anpassung wiederholt, bis das Ergebnis zufriedenstellend ist.

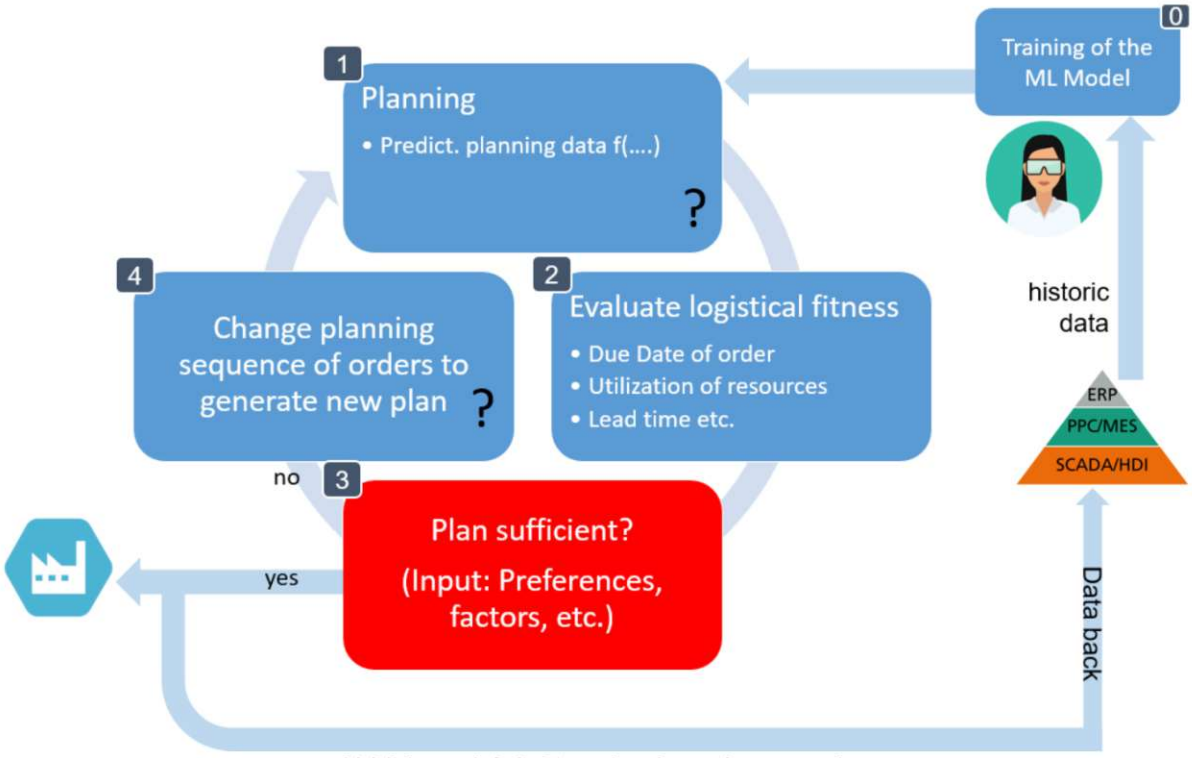

Abbildung 12: Functional- Based Approach nach Ryback et al.

Der Functional-Based Approach unterscheidet sich im Wesentlichen dadurch, dass die Planung nicht mit einem konventionellen Planungssystem, sondern bereits mithilfe der ML- Modelle durchgeführt wird. Dafür müssen zunächst die Vorhersagemodelle in Schritt 0 erstellt und mithilfe historischer Daten antrainiert werden. Nun beginnt der Iterative Zyklus mit den Schritten 1 bis 4. Der Produktionsplan wird mithilfe der aktuellen Daten durch die Machine Learning Prognosemodelle ermittelt. Anschließend wird die logistische Fitness des Produktionsplans mithilfe der KPIs ermittelt, wobei die Zielwerte des Unternehmens als Benchmark herangezogen werden. Der PQI wird hier nicht mehr herangezogen, da für den gesamten Zyklus die Planungsqualität als sehr hoch angenommen wird. Wird der Plan als nicht zufriedenstellend erachtet, werden im Schritt 4 Änderungen bezüglich der Planungssequenz durchgenommen. Die Schritte

werden so oft wiederholt, bis der Planer mit den Ergebnissen zufrieden ist und die Daten an das ERP-System übermittelt werden können.[75]

3.5 Zusammenfassung: Stand der Technik

Als Ergebnis der State of the Art Analyse lässt sich festhalten, dass in der Literatur viele verschiedene SML – Algorithmen zur Vorhersage von Zielgrößen in der Produktion verwendet werden. Eine Auflistung aller in der Literatur verwendeten Methoden ist aufgrund der großen Vielfalt an unterschiedlichen Algorithmen nicht mehr möglich. Es gibt jedoch sehr häufig eingesetzte SML - Algorithmen, die in der Literatur zu zufriedenstellenden Ergebnissen geführt haben. Ziel dieses Kapitels war es, einen Überblick über den Stand der Technik von SML in der PPS zu geben. Dazu wurden einige Studien ausgewählt und analysiert. Anschließend wurden die wichtigsten Modelle, die in der PPS zur Ermittlung von Zielgrößen verwendet werden, herausgearbeitet. Eine genaue Auflistung der in den Studien verwendeten Algorithmen finden sich in Tabelle 2.

Abkürzung	Beschreibung	Abkürzung	Beschreibung
ANN	Neuronale Netzwerke	**MARS**	Multivariate adaptive Regressionssplines
AutoML	Auto Machine Learning Modell	**MLR**	Mehrfache lineare Regression
EN	Elastisches Netz	**PR**	Polynomiale Regression
GBM	Gradient Boosting Modell	**RF**	Random Forest Modelle
kNN	k-Nächster Nachbar	**RR**	Ridge Regression
KRR	Kernel Ridge Regression	**RT**	Regressionsbaummodelle
LM	Lineare Regression	**SGD**	Stochastic Gradient Descent
LR	Lasso Regression	**SVM**	Support Vector Machines

Tabelle 2: In der Literatur verwendete SML-Algorithmen zur Vorhersage von Zielgrößen in der PPS[76]

Alle ausgewählten Studien haben gemeinsam, dass sie für die Vorhersage der Zielgrößen Regressionsmodelle verwendet haben. Zudem wurden in allen Studien die Daten zuvor mithilfe verschiedener Methoden, wie Feature Engineering und Feature

[75] vgl. Ryback et al., 2019
[76] Eigene Darstellung

Selection, vorbereitet. Alle Autoren haben mehrere Prognosemodelle aus dem Machine Learning herangezogen. In jeder Studie wurde zumindest ein lineares Modell für die Prognose verwendet. Regressionsbaummodelle haben in drei der fünf Studien zu den besten Ergebnissen führen können. Die linearen Modelle haben in den meisten Fällen ebenfalls akzeptable Ergebnisse erzielt. Für den Beispielansatz aus der Blechverarbeitung konnte ein Neuronales Netzwerk die höchste Planungssicherheit geben. In Tabelle 3 ist eine Gegenüberstellung in Form einer Tabelle durchgeführt worden.

Jahr	Autor	Datenvorbereitung	verwendete Modelle	Bestes Ergebnis	Industrie
2016	**Pfeiffer et al.**	Feature Selection	RT, RF, MLR	RF	Fertigungsindustrie
2018	**Lingitz et al.**	Sensititvitätsanalyse / Feature Selection	RT, RF, LM, LR, RR, SVM, MARS, kNN, ANN	RF	Halbleiterindustrie
2018	**Gyulai et al.**	schrittweise Vorwärtsauswahl kombiniert mit einem Random Forest Modell	LM, RF, RT, SVM	RF	Fließfertigung
2021	**Bender und Ovtcharova**	Label Encoder	AutoML	GBM	Fertigungsindustrie
2022	**Gahm et al.**	Dimensionsreduktion mittels PCA, Hyper Parameter Tuning	MLR, RF, RT, kNN, ANN, SVM, SGD, EN, PR	ANN	Metallindustrie: Blechverarbeitung

Tabelle 3: Vergleichende Ansätze von SML in der PPS[77]

Hinsichtlich der Definition zur Planungsqualität werden in der Literatur bisher einige wenige Ansätze diskutiert. Alle Definitionen haben gemeinsam, dass die Planungsqualität die Abweichungen des geplanten Ereignisses mit dem tatsächlichen Ereignis umfassen muss. Neben der Genauigkeit der Vorhersage wird in der Literatur auch die Planungssicherheit in die Definition einbezogen, welche die Stabilität des Plans gewährleisten soll. Im Zusammenhang mit der Planungsqualität wird auch oft

[77] Eigene Darstellung

der Begriff der robusten Planung verwendet. Einige Autoren vereinheitlichen die beiden Definitionen oder verwenden die Begriffe als Synonym.

Eine einheitliche Definition der Planungsqualität existiert bis heute jedoch nicht. Zur Implementierung der Planungsqualität mithilfe von ML in der PPS, bieten Ryback et al. bereits zwei gute Ansätze: den funktionsbasierten Ansatz, sowie einen evolutionsbasierten Ansatz. Diese unterscheiden sich in erster Linie durch die Art der Planung. Während der funktionsbasierte Ansatz die Planung durch ML-Algorithmen durchführt, wird im evolutionären Ansatz die Planung noch immer mit dem ERP-System durchgeführt und durch ML unterstützt. Mithilfe ihrer Ansätze kann die Planungsqualität in Form von KPIs ermittelt und auf dem Prinzip der kontinuierlichen Verbesserung durch Iteration erhöht werden.

4 Auswahl und systematischer Vergleich von SML-Algorithmen

4.1 Erläuterung des Problems

Ein bekannter Stahlerzeuger in Österreich produziert unter anderem Stahlplatten verschiedener Arten, Größen und Qualitäten. Für die Produktion der Platten ist eine genaue Vorhersage der Prozesszeiten unabdingbar. Die bisherige Vorhersage der Prozesszeiten, die aufgrund traditioneller Werkzeuge aus der Statistik durchgeführt wurden, führte zu mangelhaften Ergebnissen in der PPS. Mithilfe von SML sollen in diesem Kapitel geeignete SML-Algorithmen herangezogen und zur Prognose zukünftiger Prozesszeiten des Stahlherstellers angewendet werden. Anschließend werden diese hinsichtlich ihrer Planungsqualität miteinander verglichen und gegenübergestellt. Da die Planungsdaten aus dem ERP-System vom Unternehmen nicht zur Verfügung gestellt wurden, wird ein bereits entwickeltes Random Forrest - Modell als Referenz für die weitere Erarbeitung verwendet.

4.2 Datenbeschreibung

Die Produktion der Prozesse umfasst einige Prozessschritte und dazugehörige Prozesszeiten. Diese Prozesse wurden im Datensatz, der vom Unternehmen in Form einer CSV-Datei zur Verfügung gestellt wird, bereits zu einer einzigen Prozesszeit zusammengefasst. Der Datensatz beinhaltet die Produktionsdaten eines gesamten Kalenderjahres, wobei 60 Features (Spalten) die 3200 Prozesse (Zeilen) beschreiben. Nicht alle Features wurden mit Daten gefüllt und einige Spalten weisen Lücken auf. Außerdem besitzt der Datensatz unterschiedliche Datentypen. Der Großteil der Features ist von numerischer Natur. Einige der Daten sind vom Typ *Datum/Uhrzeit* oder vom Typ *Text*.

In einem ersten Schritt wurden die Daten formatiert und eine Beschreibung der einzelnen Features durchgeführt. Für einige der Features konnte keine genaue Beschreibung ermittelt werden. Eine genaue Auflistung aller Features, sowie die dazugehörige Beschreibung findet sich in Tabelle 4.

Bezeichung	Beschreibung	Bezeichnung	Beschreibung
FA_Nummer	Auftragsnummer	**EinStk**	Stück in den Prozess
Marke	Eingangsmaterial in der Fertigung	**ErzStk**	Stück aus dem Prozess heraus erzeugt z.B. mehrere Bleche aus einer Brammenschleifen
AnzBl	Anzahl der Bleche	**EinGew**	Gewicht in den Prozess (nur am Anfang werden die Brammen gewogen, danach berechnet)
Ziel länge [mm]	Ziellänge	**ErzGew**	berechnet aus den Maßen und den erzeugten Stück
Ziel_Breite	Zielbreite	**aktAusStk**	Ausschuss in Stück
Ziel dicke [mm]	Zieldicke	**aktAusGew**	Ausschuss in Gewicht (berechnet)
Durchschnitt Hitzen	Durschnittszeit: Hitzen	**aktAusUrs**	Ursache des Ausschusses
Anzahl gesamt Stiche	Anzahl der Gesamtstiche	**RA**	Rückmeldeart
Dauer Gesamtstiche [s]	Dauer der Gesamtsichte	**NA.**	Nacharbeit
Produktionsstart	Produktionsstart	**NG**	Nacharbeitsgrund
Produktionsende	Produktionsende	**NG.V**	Keine Beschreibung vorhanden
Prozesszeit	Zielgröße: Prozesszeit	**NA.erl**	Keine Beschreibung vorhanden
Qualität_x	Material des Stahls	**Ist.Plan**	Differenz zwischen geplanter und Ist-Fertigungswoche
Temp	Temperatur	**Erfaßt.am**	Datum der Rückmeldung
Auftrag	Fertigungsauftrag	**Uhrz.RM**	Uhrzeit der Rückmeldung
DateTime	Zusammensetzung aus Buchungsdatum und Buchungszeitpunkt	**SCHM**	Schmelze aus welcher die Bramme stammt
X	Keine Beschreibung vorhanden	**Rückmeld.**	Rückmeldenummer - ID der Rückmeldung
QuaVorsch	Qualitätsvorschrift	**AG**	Arbeitsgang
BuchDatum	Buchungsdatum	**Rück.Leist**	berechnete Zahl des Unternehmens, Substitut der IST-Zeitrückmeldung
Woche.Plan	Woche für den der Arbeitsschritt geplant wurde	**Lstg2.IGH.**	IST-Produktionszeit welche rückgemeldet wurde
S	Schicht	**Material**	Material des Stahls
KundAuft	Bestellnummer des Kunden	**EinDim1**	Breite - beim Eingang des Produktionsschrittes
KdA.Pos	Kundenauftragspositionsnummer	**EinDim2**	Dicke - beim Eingang des Produktionsschrittes
Vrg	Vorgangsnummer wie geplant	**EinDim3**	Länge - beim Eingang des Produktionsschrittes
ARBG	Arbeitsgang	**ErzDim1**	Breite - beim Eingang des Produktionsschrittes
ArbPlatz	Arbeitsplatz	**ErzDim2**	Dicke - beim Eingang des Produktionsschrittes
Rückmeldetext	Text der Rückmeldung	**ErzDim3**	Länge - beim Eingang des Produktionsschrittes
Qualität_y	Material des Stahls	**DateCol**	Buchungsdatum
Vollständiger.Name	Name Mitarbeiter	**TimeCol**	Buchungszeitpunkt
UK	Stopzeichen für die Produktion	**DLZ**	Durchlaufzeit seit Rückmeldung des Vorgängerprozesses

Tabelle 4: Beschreibung der Features[78]

[78] Eigene Darstellung

4.3 Referenzansatz

In einem ersten Versuch wurde mittels eines Regressionsbaummodells eine Vorhersage für die Planung erstellt. Dabei wurde der Datensatz zunächst in R vorbereitet. In einem ersten Schritt wurden dafür folgende 20 Features gezielt ausgewählt:

„Marke“, „Anzahl Bleche“, „Ziel länge [mm]“, „Ziel_Breite“, „Ziel dicke [mm]“, „Durchschnitt Hitzen“, „Anzahl gesamt Stiche“, „Dauer Gesamtstiche [s]“, „Temp“, „QuaVorsch“, „S“, „Vrg“, „ARBG“, „ArbPlatz“, „EinStk“, „ErzStk“, „EinGew“, „ErzGew“, „aktAusStk“ und „aktAusGew“.

Die kategorischen Daten der Features „Marke“, „QuaVorsch“, „S“, „Vrg“, „ARBG“ und „ArbPlatz“ wurden in einem weiteren Schritt mit einem Kodierer bearbeitet. Anschließend wurde der Datensatz im Verhältnis 70 zu 30 in einen Trainingsdatensatz und einen Testdatensatz aufgeteilt und als Microsoft Excel Datei exportiert. Schließlich wurden die exportierten Dateien in Python eingespeist und mithilfe eines Random Forrest Modells aus Skit-learn eine Vorhersage generiert. Zur Evaluierung der Modellleistung wird der RMSE herangezogen.

4.4 Auswahl geeigneter SML - Algorithmen

Um die, für die Problemstellung, geeigneten SML - Algorithmen auszuwählen, werden zum einen die Erkenntnisse der zuvor durchgeführten Literaturrecherche herangezogen. Zum anderen wird der Häufigkeit der Verwendung in der Literatur zur Auswahl berücksichtigt. Dazu wird eine Studie von Cadavid et al. aus dem Jahr 2019 herangezogen, in welcher die Autoren einen systematischen Literaturvergleich von 40 Publikationen über die Anwendung von maschinellem Lernen in der PPS durchführen. Dabei stellen die Autoren unter anderem die Frage nach den eingesetzten Methoden, die zur Implementierung von maschinellem Lernen in der Produktionsplanung angewendet werden. Das Ergebnis lässt sich in Abbildung 13 ablesen.[79]

Nach Kapitel 2.2 gibt es keinen universalen Algorithmus, der zur Lösung eines Problems verwendet werden kann. Die Auswahl des geeigneten Algorithmus hängt vielmehr von der Art des Problems, der Struktur der Daten und der Anzahl an Variablen ab. Die Ergebnisse aus Kapitel 3.2 zeigen, dass in der PPS ausschließlich Regressionsmodelle zum Einsatz kommen. Für die Datenvorbereitung wird meist Feature Engineering verwendet.

[79] vgl. Cadavid et al., 2019

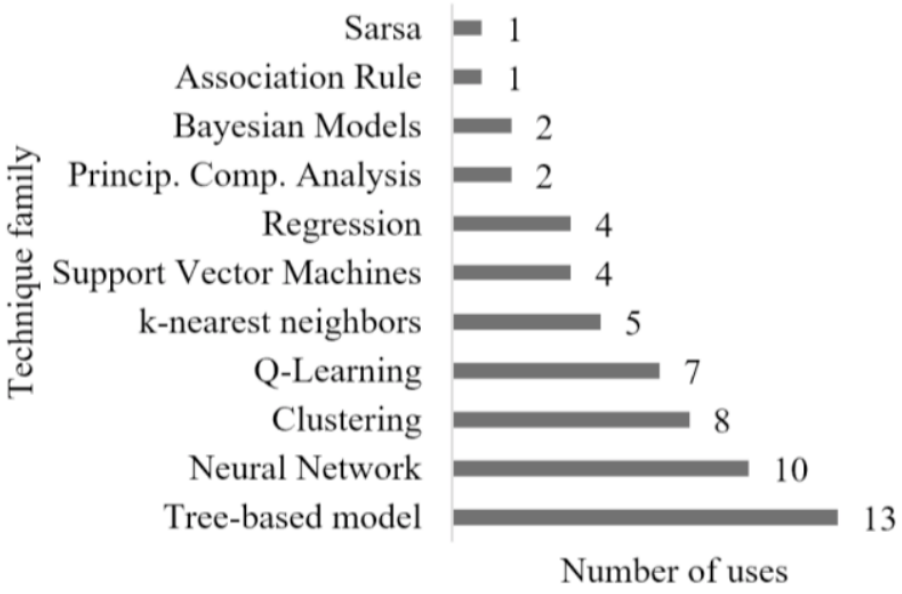

Abbildung 13: Technikfamilien und die Anzahl der Nutzungen in der Literatur[80]

Die besten Ergebnisse werden häufig durch Entscheidungsbäume erzielt, jedoch führen lineare Modelle ebenfalls zu akzeptablen Ergebnissen. Nach Abbildung 13 sind die in der Literatur meistgenutzten SML - Algorithmen Entscheidungsbaum – Modelle und Neuronale Netzwerke, gefolgt von kNN- und SVM-Algorithmen.

Nach Abwägung der oben genannten Ergebnisse werden für die Problemstellung ein lineares Modell, ein Entscheidungsbaum, ein Neuronales Netzwerk, sowie eine Support Vector Machine verwendet. Zusätzlich wird außerdem aus der Reihe der Ensemble Methoden ein Random Forest Ansatz, sowie eine Gradient Boosting Machine getestet, da diese Algorithmen in der Recherche aus Kapitel 3.2 zu den besten Ergebnissen geführt haben. Zuvor werden die Daten durch Feature Engineering vorbereitet.

4.5 Datenvorbereitung durch Feature Engineering

Bevor die SML-Modelle erstellt werden können, bedarf es einer Vorbereitung der Daten durch Feature Engineering. In diesem Kapitel wird zunächst das dafür verwendete Python Package *Feature Engine* vorgestellt, das, neben den bekannten Python Paketen *Pandas*, *Numpy und Scikit - Learn,* für diese Arbeit herangezogen wird. Anschließend wird eine Reihe von Methoden aus dem Feature Engineering erläutert und beschrieben. Dazu wird die Implementierung in Python erklärt und schließlich die Ergebnisse für den konkreten Anwendungsfall aufgezeigt.

[80] Cadavid et al., 2019

4.5.1 Feature Engine

Feature- Engine stellt eine Open-Source-Python-Bibliothek dar, die eine sehr umfassende Palette an Techniken zur Transformation der Datensätze für die Anwendung von ML-Algorithmen bietet. Die Bibliothek ist mit Scikit- Learn sowie Pandas kompatibel und ermöglicht Forschern und Datenwissenschaftlern den freien und benutzerfreundlichen Zugang zu Datentransformationsverfahren. Bei der Vorbereitung des Datensatzes können verschiedene Feature-Transformationen auf verschiedene Variablengruppen angewendet werden. Dem Benutzer ist es möglich jene Variablen auszuwählen, die transformiert werden sollen. So kann der gesamte Datensatz als Eingabe verwendet werden, jedoch werden nur die angegebenen Variablen verändert. Die Techniken von Feature -Engine sind mit allen Techniken der Python Umgebung kombinierbar.[81]

Die Python Bibliothek Feature- Engine ist über den unten angeführten Link aufrufbar und muss anschließend installiert werden. Dies geschieht über den Python Package Index (PyPi) - Installer in der Konsole durch die Eingabe des Installationsbefehls. Mit der *Fit- Methode* werden die Funktionen mit den eingegebenen Merkmalen zunächst angelernt. Anschließend werden die Variablen mit der *Transform- Methode* umgewandelt. Die Bibliothek Feature- Engine enthält folgende Transformatoren: [82]

- Imputation fehlender Daten
- Kodierung kategorischer Features
- Diskretisieren
- Entfernung von Ausreißern
- Feature - Transformation
- Erstellung neuer Features
- Feature Selection
- Datum-Zeit-Features
- Zeitreihentransformation
- Vorverarbeitung

Für Einsteiger im Feature Engineering bietet die Webseite einige Hilfemöglichkeiten an. Ein Benutzerhandbuch zeigt zusätzliche Informationen über die Feature-Engine-Transformatoren und Feature-Engine-Transformationen im Allgemeinen sowie zusätzliche Anwendungsbeispiele. Das Handbuch ist unter dem unten angeführten Link aufrufbar.[83] Für ein besseres Verständnis werden außerdem drei Online-Kurse und zwei Bücher angeboten.[84]

[81] vgl. Galli, 2021, S. 1
[82] https://feature-engine.readthedocs.io/en/latest/, (letzter Zugriff: 23.01.2023)
[83] https://feature-engine.trainindata.com/en/latest/user_guide/index.html, (letzter Zugriff: 23.01.2023)
[84] https://feature-engine.readthedocs.io/en/latest/index.html, (letzter Zugriff: 23.01.2023)

Die Datenvorbereitung des Datensatzes wird in dieser Arbeit durch die Datenbank Feature- Engine und der Zuhilfenahme klassischer Python-Bibliotheken, wie etwa Pandas, durchgeführt. Das Buch *Python Feature Engineering Cookbook* von Soledad Galli aus dem Jahr 2020 bietet eine Art Kochrezept für das Feature Engineering mit der Datenbank Feature- Engine. Die Vorgehensweise für die Datenvorbereitung wird daher in Anlehnung an das Buch durchgeführt.

4.5.2 Entfernen von Duplikaten

Ein Problem in vielen Datensätzen ist das das Vorhandensein von Duplikaten innerhalb der Beobachtungen. Auch die Anzahl der Beobachtungen eine entscheidende Rolle bei der Erstellung von ML- Modellen spielt, tragen Duplikate keinen Mehrwert für die Modelle bei. Die Behandlung der Duplikate kann dabei auf unterschiedliche Arten erfolgen. Es können etwa mehrfache Aufzeichnungen entfernt und auf eine Aufzeichnung reduziert werden oder es werden alle Duplikate gänzlich entfernt. Python bietet zur Erkennung und Eliminierung von Duplikaten die Funktion *drop_duplicates* an.[85]

Innerhalb des Datensatzes wurden einige Beobachtungen doppelt angeführt. Diese Beobachtungen müssen in einem ersten Schritt entfernt werden. Dazu wird zunächst der Datensatz mithilfe von Pandas *read_excel* - Funktion geladen.[86]

Schließlich werden die Duplikate aus dem Datensatz mithilfe der erwähnten Funktion entfernt.[87] Durch den Vorgang wird der Datensatz auf 2930 Beobachtungen reduziert. Folglich wurden 270 Duplikate entfernt. Als Referenzspalte wird das Feature „FA_Nummer“ herangezogen, welches die Auftragsnummer repräsentiert. Der Parameter *keep* beschreibt, ob und welches Duplikat erhalten bleiben soll. Es besteht die Möglichkeit alle Beobachtungen zu entfernen, oder die zuerst Gesehene oder zuletzt Gesehene Beobachtung zu behalten. In diesem Fall werden alle Duplikate entfernt, mit Ausnahme der zuletzt Gesehenen.[88] Der zugehörige Quellcode findet sich in Anhang 7.1.1.

4.5.3 Eliminierung unnützer Features

In einem ersten Schritt werden aus dem Datensatz einige Features im Vorfeld entfernt, die für die Erstellung der Modelle mit großer Sicherheit keinen Mehrwert liefern werden. Die Voreliminierung unnützer Features wird in diesem Fall manuell

[85] vgl. Sarkar et al., 2018, S. 147
[86] vgl. https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html, (letzter Zugriff: 23.01.2023)
[87] vgl. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html, (letzter Zugriff: 23.01.2023)
[88] vgl. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html, (letzter Zugriff: 23.01.2023)

durchgeführt, da es ein grundlegendes Verständnis für den Datensatz erfordert. Durch automatisierte Feature Eliminierung können nach Kapitel 2.3 wertvolle Features verloren gehen. Zunächst werden jene Merkmale entfernt, die lediglich eine Nummerierung der Prozesse, beziehungsweise der Aufträge, beschreiben. Darunter fallen die Features „FA_Nummer“, „KundAuft“ und „Auftrag“.

Im nächsten Schritt werden jene Merkmale entfernt, die ein Datum oder eine Uhrzeit beinhalten. Dadurch werden die Features „Woche.Plan“, „Produktionsstart“, „Produktionsende“, „DateTime“, „BuchDatum“, „Erfaßt.am“, „Uhrz.RM“, „DateCol“ und“ TimeCol“ im Vorfeld aus dem Datensatz eliminiert. Das Merkmal „Vollständiger.Name“ beinhaltet den Namen eines Mitarbeiters. Das Feature „RA“ gibt die Art der Rückmeldung an. Diese beiden Features werden ebenfalls entfernt. Zuletzt werden zwei Features entfernt, die mit einem Dritten Merkmal ident sind. Dies betrifft die Features „Qualität_x“ und „Qualität_y“, da sie denselben Inhalt wie das Feature „Marke“ beinhalten. Die Eliminierung der Features erfolgt in Python mit der Pandas Funktion *drop*.[89] Die Implementierung in Python ist dem Anhang 7.1.2 zu entnehmen.

4.5.4 Quantifizierung fehlender Daten

Fehlende Daten in den einzelnen Beobachtungen kommen in den meisten Datensätzen vor und stellen ein Problem für die Modellierung dar. Das Python Package Scikit- Learn, das zur Erstellung der ML- Modelle verwendet wird, unterstützt keine fehlenden Werte als Eingabe für die Lernmodelle. Daher müssen fehlende Daten zunächst identifiziert und anschließend bearbeitet werden. Dafür lohnt es sich, zunächst eine Quantifizierung fehlender Daten in den einzelnen Features vorzunehmen. Dies geschieht in Python mit einer einfachen Programmierung wie folgt:[90]

Zunächst wird der prozentuelle Anteil an fehlenden Werten für alle Features ermittelt. Anschließend wird ein prozentueller Wert festgelegt, der die maximale Anzahl fehlender Werte festlegen soll. In diesem Fall wurde festgelegt, dass maximal 5 Prozent der Daten fehlen dürfen.

In Abbildung 14 ist die Quantifizierung der Features in Form eines Balkendiagramms dargestellt. Die Y- Achse beschreibt dabei die Anzahl der fehlenden Werte, innerhalb des jeweiligen Merkmals, in Prozent.

[89] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html, (letzter Zugriff: 23.01.2023)
[90] vgl. Galli, 2022, S. 15

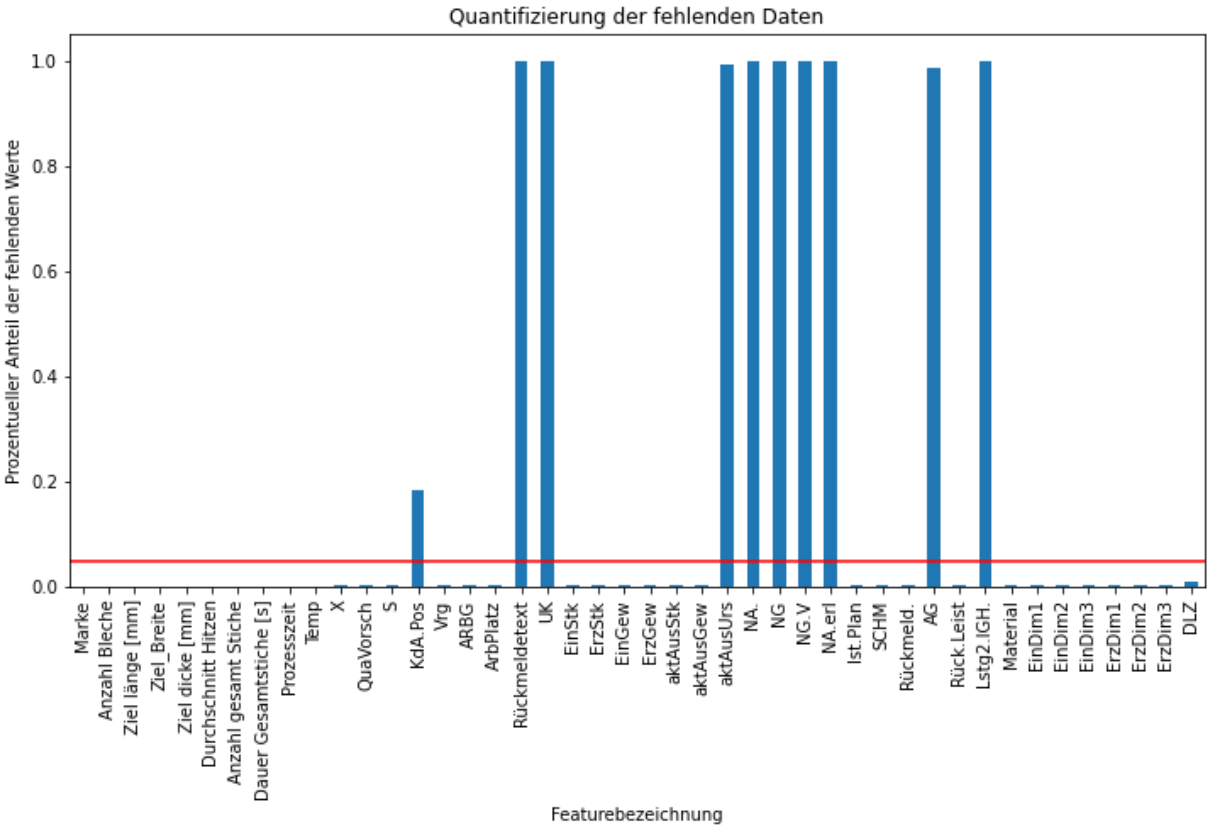

Abbildung 14: Quantifizierung fehlender Daten[91]

Mit einer einfachen *for* - Schleife werden nun alle Features aus dem Datensatz entfernt, die mehr als 5 Prozent an fehlendem Daten aufweisen. Durch die Schleife werden die Features „KdA.Pos“, „Rückmeldetext“, „UK“, „aktAusUrs“, „NA. „, „NG“, „NG.V“, „NA.erl“, „AG“ und „Lstg2.IGH“ entfernt.

Da der Datensatz nach wie vor fehlende Werte beinhaltet, werden im letzten Schritt alle Beobachtungen, die zumindest einen fehlenden Wert (Not a Number - NaN) beinhalten, entfernt. Dazu wird die Pandas Funktion *dropna* herangeozgen.[92] Durch den Vorgang wird die Anzahl der Beobachtungen auf 2893 reduziert und 37 Reihen werden entfernt. Der Quellcode aus Python zur Quantifizierung der fehlenden Daten ist in Anhang 7.1.4 zu finden.

4.5.5 Identifizierung kategorischer und numerischer Features

Wie bereits in Kapitel 2.3.1 erwähnt, können Merkmale unterschiedliche Datentypen und Werte beinhalten. So können die Features kategorischer oder numerischer Natur sein und sie können im Allgemeinen diskrete oder kontinuierliche Werte beinhalten. Diskrete Variablen sind in der Regel vom Typ *Integer* (*int*). Kontinuierliche Variablen weisen meist den Datentyp *Float* (*float64*) auf. Kategorische Features werden in

[91] Eigene Darstellung
[92] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html, (letzter Zugriff: 23.01.2023)

Pandas meist mit dem Typ *object* gekennzeichnet. Allerdings funktioniert diese Kategorisierung nicht immer. Daher muss der Datentyp manuell betrachtet und gegebenfalls angepasst werden.[93]

Feature	Datentyp	Änderung erfoderlich	Feature	Datentyp	Änderung erfoderlich
Marke	object		EinStk	float64	
Anzahl Bleche	int64	x	ErzStk	float64	
Ziel länge [mm]	int64	x	EinGew	float64	
Ziel_Breite	int64	x	ErzGew	float64	
Ziel dicke [mm]	float64		aktAusStk	float64	
Durchschnitt Hitzen	int64	x	aktAusGew	object	x
Anzahl gesamt Stiche	int64	x	Ist.Plan	float64	
Dauer Gesamtstiche [s]	float64		SCHM	object	
Prozesszeit	float64		Rückmeld.	object	x
Temp	int64	x	Rück.Leist	object	x
X	float64		Material	object	x
QuaVorsch	float64	x	EinDim1	object	x
S	object		EinDim2	float64	
Vrg	object		EinDim3	object	x
ARBG	object		DLZ	float64	
ArbPlatz	float64	x			

Tabelle 5: Datentypen der Features vor der Korrekturmaßnahme[94]

Mithilfe der *dtypes*- Funktion aus Pandas werden zunächst die Datentypen der Features erfasst.[95] (Siehe Anhang 7.1.5) Tabelle 5 gibt einen Überblick über die Datentypen der einzelnen Features. Im Reiter „Änderung erforderlich“ werden alle Features markiert, bei denen die automatische Erkennung des Datentyps nicht funktioniert hat und eine Änderung des Datentyps durchgeführt werden muss.

Jene Features, die nach Betrachtung der Daten manuell markiert wurden, werden nun entsprechend ihrem Datentyp geändert. So werden die Merkmale „Marke“, „QuaVorsch“, „S“, „Vrg“, „ARBG“, „ArbPlatz“ und „SCHM“ als kategorische Features dem Datentyp *object* zugeschrieben. Allen anderen Features, die von numerisch-kontinuierlicher Natur sind, wird der Datentyp *float64* zugeordnet.

[93] vgl. Galli, 2022, S. 11
[94] Eigene Darstellung
[95] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dtypes.html, (letzter Zugriff: 23.01.2023)

4.5.6 Training- Test Split

Ein häufiger Fehler bei der Leistungsbestimmung eines ML-Modells liegt im falschen Training der Modelle. Das Modell muss mit einem gesonderten Trainingsdatensatz angelernt werden. Die verwendeten Trainingsdaten, die dem Modell in seiner Lernphase übergeben werden, dürfen in der Testphase keinen Gebrauch mehr finden. Ansonsten würde das Modell hervorragend abschließen, jedoch falsche Ergebnisse hinsichtlich seiner Vorhersagegenauigkeit liefern. Aus diesem Grund muss ein sogenannter Hold-out-Datensatz herangezogen werden. Dieser beschreibt eine willkürliche Stichprobe des Datensatzes, der zuvor entnommen wird und anschließend dem Modell bei der Evaluierung seiner Leistung übergeben wird.[96] Dazu wird der Datensatz in einem ersten Schritt (siehe Kapitel 2.2.1) in einen Eingabevektor und einen Zielvektor aufgeteilt. Dies geschieht erneut mit der Pandas Funktion *drop*. X stellt den Eingabevektor dar und Y den Ausgabevektor. Die Zielgröße ist in dieser Arbeit der Vektor „Prozesszeit“.

Mit der Training- Test- Split Funktion aus Skit-Learn werden die Datensätze anschließend in einen zufälligen Trainings- und Testdatensatz aufgeteilt.[97] (siehe Anhang 7.1.5) Zur Unterteilung der Daten wird ein üblicher 70- 30 Ansatz gewählt, wie er etwa bei Sarkar et al. oder Galli verwendet wird. Somit werden 70 Prozent der Daten als Trainingsdatensatz und 30 Prozent der Daten als Testdatensatz generiert.

4.5.7 Umgang mit Ausreißern

Als Ausreißer werden Datenpunkte bezeichnet, die sich signifikant von den übrigen Daten unterscheiden. Diese Ausreißer können die Leistung vieler Modelle des maschinellen Lernens beeinträchtigen, wie etwa die lineare Regression. Aus diesem Grund ist es sinnvoll, Ausreißer im Datensatz zu bearbeiten. Eine Möglichkeit, mit Ausreißern umzugehen, besteht darin, die Ausreißer als fehlende Informationen zu behandeln und zusammen mit den übrigen fehlenden Daten zu ergänzen. Dafür muss allerdings eine Imputation der fehlenden Daten vorgenommen werden.[98]

Weiters ist es möglich, eine Begrenzung der Variablen hinsichtlich eines Intervalls vorzunehmen. Beim *Winsorizing* werden die Daten so transformiert, dass die Extremwerte auf einen bestimmten Wert begrenzt werden, der näher am Mittelwert der Verteilung liegt. Im Unterschied zum *Trimming* werden beim Winsorizing die Beobachtungen nicht entfernt, sondern die Extremwerte ausgetauscht. Eine typische Strategie besteht darin, Ausreißer auf ein bestimmtes Perzentil zu setzen. Beim 90%-Winsorizing werden beispielsweise alle Daten unterhalb des 5. Perzentils auf den Wert

[96] vgl. Johnston und Mathur, 2019, S. 280 ff
[97] vgl. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html, (letzter Zugriff: 23.01.2023)
[98] vgl. Galli, 2022, S. 196 f

des 5. Perzentils und alle Daten oberhalb des 95. Perzentils auf den Wert des 95. Perzentils gesetzt. Beim sogenannten *Capping* werden, ähnlich wie beim *Winsorizing,* ein Maximalwert sowie ein Minimalwert als Grenze festgelegt. Beim Capping werden so die Ausreißer durch Werte ersetzt, die näher an den Grenzwerten der Variablen liegen. Im Normalfall werden diese Grenzen durch den Mittelwert plus oder minus der Standardabweichung oder der Quantile-Methode bestimmt.[99]

Zuletzt können die Ausreißer auch entfernt werden. Dieser Vorgang wird als *Trimmen* oder *Trunkieren* von Beobachtungen bezeichnet. Dabei gibt es gängige Methoden, wie das Trimmen der Daten durchgeführt werden kann. Sofern die Variable normalverteilt ist, werden die Grenzen des Trimmers durch den Mittelwert plus oder minus der zwei- bis dreifachen Standardabweichung bestimmt, da etwa 99 Prozent der Daten zwischen diesen Grenzen verteilt sind.[100]

Für die Vorliegende Arbeit wird die Methode des Trimmens verwendet. Der Outlier-Trimmer von Feature-Engine entfernt Beobachtungen mit Ausreißern aus dem Datensatz. Dazu berechnet er zunächst den Höchst- und den Mindestwert, bei deren Überschreitung ein Wert als Ausreißer gilt und entfernt diesen anschließend. Mit der Trimmer-Methode wird für den Datensatz ein empfohlenes 90 - Percentil als rechte Grenze festgelegt.[101]

Als Variable wurde dem Trainingssatz für den Vorgang der Trimmung ein neues Feature hinzugefügt. Das Feature „Prozesszeit/Outlier“ enthält die Werte der Zielgröße“Prozesszeit“. Die Trimmung muss sowohl für den Eingangsvektor als auch den Ausgangsvektor durchgeführt werden.

Nachdem die Trimmung durchgeführt wurde, wird das neue Feature „Prozesszeit/Outlier“ wieder entfernt, da es im Eingangsvektor nicht vorkommen darf.

[99] vgl. Galli, 2022, S. 202ff
[100] vgl. Galli, 2022, S. 196 f
[101] vgl. https://feature-engine.readthedocs.io/en/latest/api_doc/outliers/OutlierTrimmer.html, (letzter Zugriff: 23.01.2023)

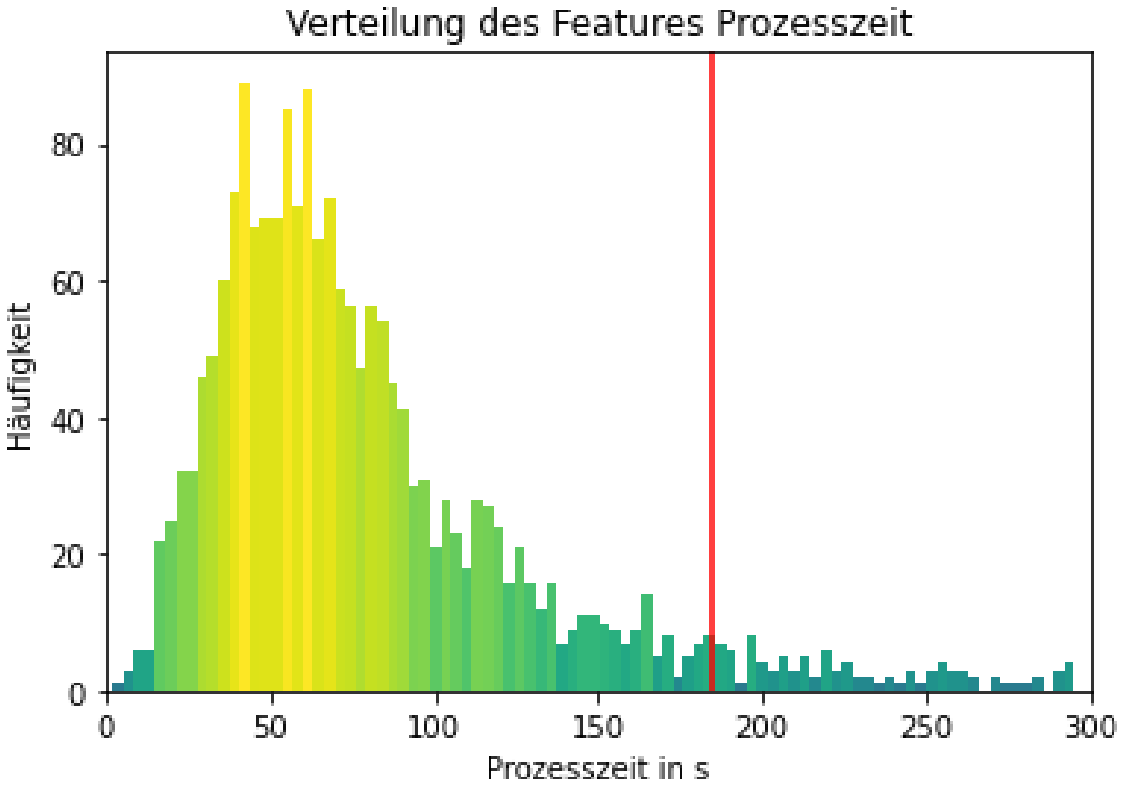

Abbildung 15: Verteilung der Werte des Features „Prozesszeit“[102]

Abbildung 15 zeigt die Verteilung der Werte des Features „Prozesszeit“. Die rote Linie zeigt den berechneten Schwellenwert der Funktion. Alle Beobachtungen, deren Werte über diesem Schwellenwert liegen, werden aus dem Datensatz entfernt. Der zugehörige Quellcode findet sich in Anhang 7.1.6.

4.5.8 Kategoriale Kodierer

4.5.8.1 Bestimmung der Kardinalität

Die Anzahl der eindeutigen Kategorien in einem kategorischen Merkmal wird als *Kardinalität* bezeichnet. Beinhaltet eine Variable beispielsweise Geschlechter, die lediglich die Werte weiblich und männlich annehmen kann, so beträgt die Kardinalität 2. Mithilfe der *nunique*- Funktion aus Pandas[103] wird die Kardinalität der kategorischen Features „Marke", „QuaVorsch", „S", „Vrg", „ARBG", „ArbPlatz" und „SCHM" ermittelt und in Abbildung 16 dargestellt.[104]

[102] Eigene Darstellung
[103] vgl. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.nunique.html, (letzter Zugriff: 23.01.2023)
[104] vgl. Galli, 2022, S. 18

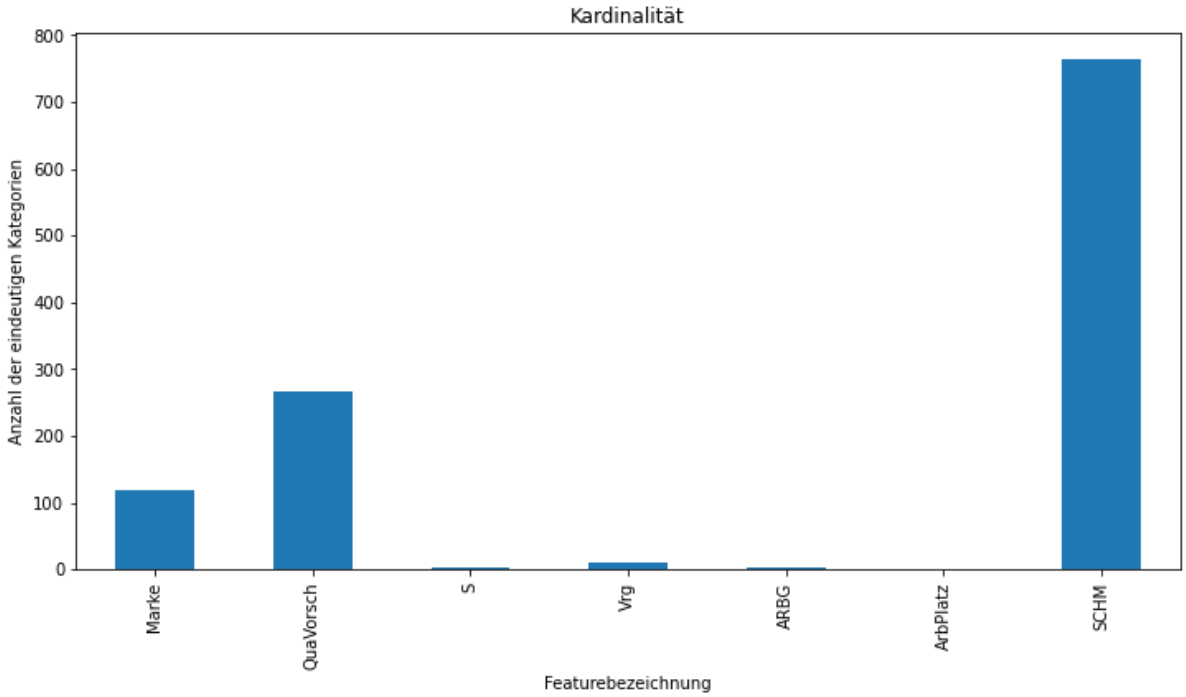

Abbildung 16: Kardinalität der kategorischen Variablen[105]

Dabei wird ersichtlich, dass die Merkmale „Marke“, „QuaVorsch“ und „SCHM“ sehr viele Kategorien aufweisen. Um diese Kategorien zu vereinfachen, wird im nachfolgenden Kapitel eine Kodierung von seltenen Werten aus dem Package Feature- Engine angewendet.

4.5.8.2 Kodierung von seltenen Kategorien

Die unterschiedlichen Werte eines kategorischen Merkmals erscheinen in einem Feature mit unterschiedlicher Häufigkeit. Einige Kategorien treten meist häufiger auf, während andere Kategorien in nur sehr wenigen Beobachtungen vorkommen. Als *Rare Labels* (zu deutsch: seltene Werte) werden Kategorien bezeichnet, die in einem nur sehr kleinen Prozentsatz der Beobachtungen vorkommen. Eine generelle Faustregel zur Bestimmung dieser Rare Labels existiert nicht. In vielen Fällen werden Werte unter 5 Prozent als selten angesehen. Rare Labels kommen häufig nur in der Trainingsmenge oder nur im Testdatensatz vor. Dadurch können ML-Algorithmen anfälliger für eine Überanpassung werden oder einzelne Beobachtungen können nicht mehr bewertet werden. Um die Komplikationen zu vermeiden, kann ein *Rare Label Encoder* verwendet werden, der diese seltenen Werte in einer neuen Kategorie zusammenfasst.[106]

Der Rare-Label-Encoder aus dem Paket von Feature- Engine kategorisiert seltene Werte in einer neuen Kategorie mit der Bezeichnung „Rare“. So werden alle Werte

[105] Eigene Darstellung
[106] vgl. Galli, 2022, S. 129

eines Merkmals, deren Häufigkeit unter dem Toleranzwert liegen, in die Kategorie „Rare“ eingeordnet und ersetzt.[107]

Nach Anleitung des Handbuchs wird der Rare-Label-Encoder zunächst auf die Trainingsdaten mit der üblichen Fit-Methode angewendet. Er berücksichtigt dabei ausschließlich die Variablen „Marke“, „QuaVorsch“ und „SCHM“. Anschließend wird die gleiche Transformation, mit der Transform-Methode, auch auf den Testdatensatz angewendet. Als Toleranzwert werden iterativ mehrere Werte zwischen 0.1 Prozent und 5 Prozent ausprobiert. Als endgültigen Wert wird der Wert 0.1 Prozent herangezogen.[108] (siehe Anhang 7.1.7.1)

Als Ergebnis der Transformation reduziert sich die Kardinalität und somit die Anzahl der Kategorien in den drei ausgewählten Features. Die Ergebnisse sind Abbildung 17 zu entnehmen. Der blaue Balken zeigt die Kardinalität vor der Transformation. Der rote Balken bildet die Kardinalität nach der Transformation ab.

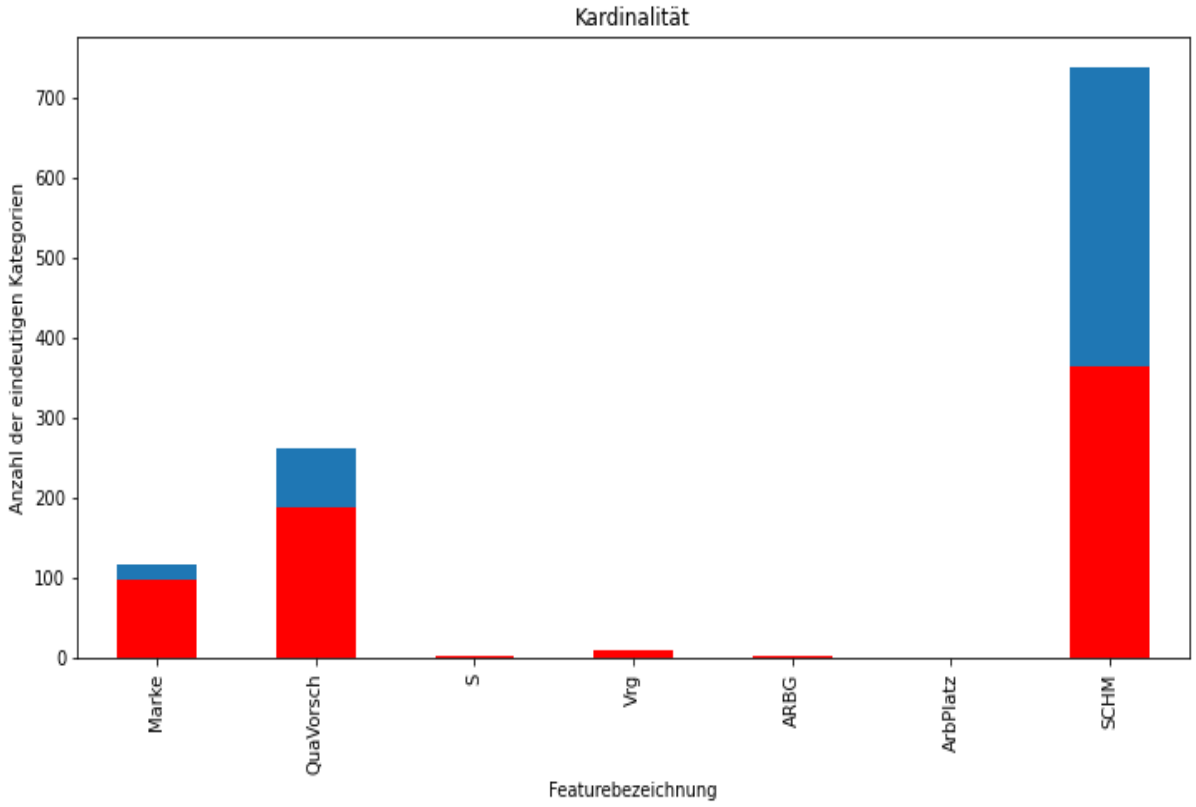

Abbildung 17: Änderung der Kardinalität der ausgewählten Features[109]

[107] vgl. https://feature-engine.readthedocs.io/en/latest/api_doc/encoding/RareLabelEncoder.html, (letzter Zugriff: 24.01.2023)
[108] vgl.https://feature-engine.readthedocs.io/en/latest/user_guide/encoding/RareLabelEncoder.html#rarelabel-encoder, (letzter Zugriff: 24.01.2023)
[109] Eigene Darstellung

4.5.8.3 Ordinale Kodierung

Kategoriale Variablen beinhalten häufig *Strings,* wie etwa die Namen der Marke. Die Open-Source-Python-Bibliothek Scikit- Learn unterstützt diese Werte für ihre ML-Algorithmen nicht. Daher müssen diese Strings zunächst in numerische Daten umgewandelt werden. Dieser Vorgang wird kategoriale Kodierung genannt.[110] Im ersten Moment bietet es sich an, jeder Kategorie innerhalb des Merkmals einen numerischen Wert zuzuordnen, etwa von 1 bis k. Dabei könnte das Modell allerdings versuchen, die Reihenfolge als reine numerische Werte zu interpretieren. Sofern das Merkmale keine ordinale Variable darstellt, bei der die Reihenfolge der Werte eine Rolle spielt, wäre eine Interpretation des Modells falsch und könnte zu einer schlechteren Performance führen.[111]

Feature Engine bietet eine Reihe von kategorialen Kodierern an. In Abbildung 18 ist eine Übersicht dieser Kodierer aufgelistet, wobei ersichtlich ist, dass nicht jeder Kodierer für das konkrete Problem geeignet ist. Für diese Arbeit wird ein Count-Frequency- Encoder verwendet, da er für ein Regressionsproblem herangezogen werden kann. Dieser Kodierer ersetzt die Kategorien durch die Anzahl der Beobachtungen in der jeweiligen Kategorie. Erscheint beispielsweise der Wert in zehn Beobachtungen, so wird für diese Kategorie der Wert „10“ verwendet.[112]

	Monotonic encoding	Suitable regression	Suitable Binary Classification	Suitable Multi-class Classification	Description
OneHotEncoder	√	√	√	√	Creates dummy/ binary variables from every category
CountFrequencyEncoder	×	√	√	√	Replaces categories by the count or frequency of observations
OrdinalEncoder	√ if ordered by target, × otherwise	√	√	√ if numbers assigned arbitrarily × otherwise	Replaces categories by integers arbitrarily or ordered by target mean value
MeanEncoder	√	√	√	× The transformer will return a value, but the mean of different classes does not have mathematical sense	Replaces categories by the target mean value per category
WoEEncoder	√	×	√	×	Replaces categories by the Weight of Evidence (WoE)
PRatioEncoder	√	×	√	×	Replaces categories by a ratio of probabilities
DecisionTreeEncoder	√	√	√	√	Replaces categories by the predictions of a decision tree
RareLabelEncoder	NA	√	√	√	Groups infrequent categories into a new category

Abbildung 18: Kodierer in Feature- Engine[113]

Die Kodierung erfolgt erneut nach dem Handbuch, wobei zunächst der Kodierer durch den Trainingsdatensatz antrainiert wird. Anschließend wird sowohl der Trainingsdatensatz als auch der Testdatensatz transformiert. Da es vorkommen kann,

[110] vgl. Galli, 2022, S. 92
[111] vgl. Sarkar et al., 2018, S. 203
[112] vgl. https://feature-engine.readthedocs.io/en/latest/api_doc/encoding/CountFrequencyEncoder.html, (letzter Zugriff: 24.01.2023)
[113] https://feature-engine.readthedocs.io/en/latest/user_guide/encoding/index.html, (letzter Zugriff: 24.01.2023)

dass der Testdatensatz neue Kategorien beinhaltet, die im Testdatensatz nicht vorgekommen sind, müssen diese Beobachtungen aus dem Testdatensatz entfernt werden. In dieser Arbeit werden dadurch keine Beobachtungen entfernt. (Siehe Anhang 7.1.7.2)

4.5.9 Feature Selection

In Kapitel 2.3.3 werden die gängigen Strategien zur Auswahl von Merkmalen beschrieben. Dabei stehen einige gute Gründe im Raum, eine Feature Selection (zu deutsch: Feature Auswahl) durchzuführen. Zum einen kommt es zu einer Reduzierung der Trainingszeit und, damit verbunden, zu einer Reduzierung der Berechnungszeit im Rahmen der Modellerstellung. Weiters führt eine Dimensionsreduktion zu einer geringeren Komplexität. Dadurch können die Ergebnisse der Modelle besser interpretiert werden (vergleiche Kapitel 2.3).

Die Feature-Selektionstransformatoren von Feature Engine versuchen Merkmale zu identifizieren, die eine geringe Vorhersagekraft haben und entfernen diese aus dem Datensatz. Die meisten der von Feature Engine unterstützten Algorithmen zur Auswahl von Features sind in anderen Bibliotheken noch nicht verfügbar. Diese Algorithmen wurden bei Data-Science-Wettbewerben gesammelt oder in der Industrie eingesetzt.

Die Auswahl der Features basiert dabei auf mehreren Strategien. Zum einen können die Features auf Grundlage ihrer intrinsischen Eigenschaften, wie ihrer Verteilung, ausgewählt werden. Eine weitere Strategie besteht darin, eine Beziehung zwischen den Merkmalen zu identifizieren. So können etwa stark korrelierte oder doppelte Features entfernt werden. Zuletzt können Merkmale auch auf der Grundlage ihrer Beziehung zur Zielgröße ausgewählt werden. Einige Verfahren zur Auswahl von Merkmalen beinhalten auch das Training von ML-Modellen und die Auswahl anhand der Modell- Performance von Features.[114]

Für die Auswahl der Features wurden in dieser Arbeit mehrere der zur Verfügung stehenden Transformatoren iterativ angewendet und ihre Performance begutachtet. Im Folgenden wird auf die im letzten Iterationsschritt ausgewählten Selektoren eingegangen.

4.5.9.1 Auswahl der Features nach ihrer Einzelleistung

Der Transformator *Select By Single Feature Performance* wählt Merkmale auf Basis der Leistung eines zuvor ausgewählten ML-Modells aus, das mit dem einzelnen Feature antrainiert wurde. So wird für jedes einzelne Merkmal aus dem Datensatz ein eigenes ML-Modell antrainiert und schließlich die Leistung des Modells getestet. Die Modelle werden dabei mithilfe einer Kreuzvalidierung antrainiert. Die Anzahl der

[114]vgl. https://feature-engine.readthedocs.io/en/latest/user_guide/selection/index.html, (letzter Zugriff: 24.01.2023)

Kreuzvalidierung sowie das zu trainierende ML-Modell kann vom Benutzer selbst festgelegt werden. Auf Basis der Performance der ML-Modelle wird dann die Selektion der Features durchgeführt. Dafür wird ein zuvor definierter Schwellenwert festgelegt. Wenn dieser überschritten wird, wird das Feature aus dem Datensatz entfernt, andernfalls behalten.[115]

Um die Performance der einzelnen Features zu ermitteln, wird ein lineares Regressionsmodell aus der Bibliothek sklearn von Sckit-Learn verwendet.[116] Dieses wird dem Parameter *estimator* in der Funktion zugeordnet. Weitere Informationen zur linearen Regression sind dem Kapitel 4.6.2 zu entnehmen.

Zur Evaluierung der Leistung der ML-Modelle wird mit dem Parameter *scoring* der RMSE von *sklearn* herangezogen.[117] Die Modelle werden mithilfe einer 10-fachen Kreuzvalidierung antrainiert. Der Parameter *threshold* beschreibt den Schwellenwert für die Auswahl der Features. Wird dieser Parameter auf *None* gesetzt, werden die Features ausgewählt, deren Leistung über dem Durchschnittswert der Leistungen aller Features liegt.[118] (siehe Anhang 4.5.9.1)

Schließlich wird der Trainingsdatensatz mithilfe der Fit- Transform- Funktion transformiert. Dazu werden dem Transformator sowohl der Eingabevektor als auch der Zielvektor übergeben. Nach der Dimensionsreduktion werden die entfernten Features auch aus dem Testdatensatz entfernt.

[115] vgl. https://feature-engine.readthedocs.io/en/latest/api_doc/selection/SelectBySingleFeaturePerformance.html#feature_engine.selection.SelectBySingleFeaturePerformance, (letzter Zugriff: 24.01.2023)
[116] vgl. https://scikit-learn.org/ stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression, (letzter Zugriff: 25.01.2023)
[117] vgl. https://scikit-learn.org/stable/modules/model_evaluation.html, (letzter Zugriff: 25.01.2023)
[118] vgl. https://feature-engine.readthedocs.io/en/latest/user_guide/selection/SelectBySingleFeaturePerformance.html#single-feat-performance, (letzter Zugriff: 25.01.2023)

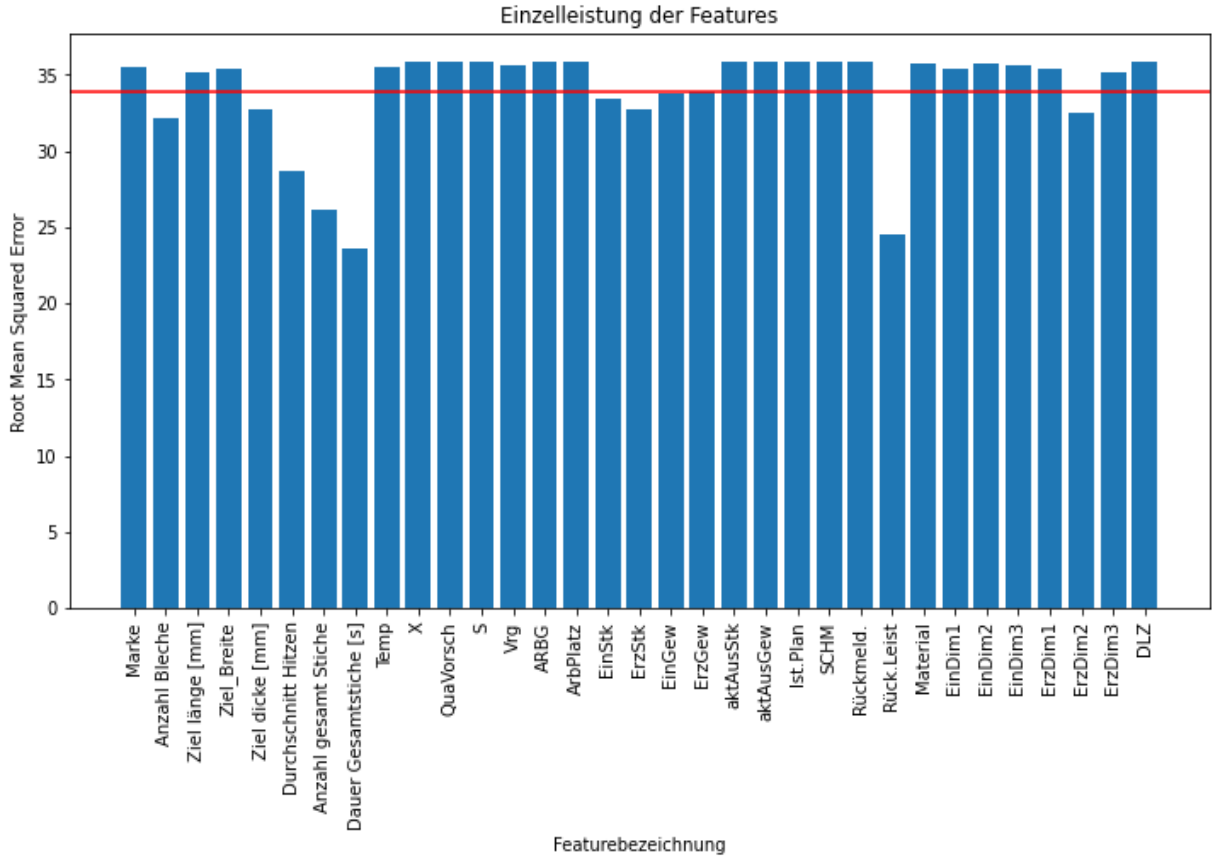

Abbildung 19: Einzelleistung der einzelnen Features[119]

Abbildung 19 zeigt die Einzelleistungen aller Features, die durch das lineare Regressionsmodell berechnet wurden. Die rote Linie zeigt den Mittelwert aller Einzelleistungen, der dem Schwellenwert für die Feature Auswahl entspricht. Alle Features deren RMSE-Wert über dem Schwellenwert liegen, werden durch den Transformator aus dem Datensatz entfernt. Durch den Vorgang werden etwa die Features "Marke", „Ziel länge [mm]", „Ziel_Breite", „Temp", „X", „QuaVorsch", „S", „Vrg", „ARBG", „ArbPlatz", „aktAusStk", „aktAusGew", „Ist.Plan", „SCHM", „Rückmeld.", „Material", „EinDim1", „EinDim2", „EinDim3", „ErzDim1", „ErzDim3" und „DLZ" entfernt.

4.5.9.2 Rekursive Feature Eliminierung

Mit der Hilfe eines weiteren Transformators aus dem Paket Feature Engine, dem Recursive-Feature-Elimination, wird eine weitere Feature Auswahl durch ein rekursives Eliminierungsverfahren vorgenommen. Dieses Verfahren wird dazu verwendet, um Abhängigkeiten der Features zu beseitigen, die im Modell bestehen können. Der Prozess läuft dabei folgendermaßen ab: [120]

[119] Eigene Darstellung
[120] vgl. https://feature-engine.readthedocs.io/en/latest/user_guide/selection/RecursiveFeatureElimination.html#recursive-elimination, (letzter Zugriff: 24.01.2023)

1. Zunächst wird ein ML-Modell (*estimator*) unter Anwendung aller vorhandenen Features antrainiert.
2. Nun werden für jede Variable Wichtigkeitswerte berechnet und anschließend nach ihrer Wichtigkeit geordnet. Dies geschieht mithilfe der Funktion *feature_importances_* oder *coef_* aus Pandas. Die Werte werden im Hintergrund gespeichert und es wird zu Schritt 3 übergegangen.
3. Das am wenigsten wichtige Feature wird entfernt. Nun wird das Modell erneut antrainiert und die Leistung mithilfe des Parameters *scoring* evaluiert.
4. Der Leistungsunterschied zwischen den beiden Modellen wird ermittelt.
5. Liegt der Unterschied der Leistung über dem Schwellenwert *threshold*, so wird das Feature aus dem Datensatz entfernt.
6. Die Schritte 3 – 5 werden so lange wiederholt, bis alle Features bewertet wurden.

Das Training des Modells und die Leistungsbewertung werden mit einer 10-fachen Kreuzvalidierung durchgeführt. Die Leistung wird mit dem RMSE gemessen. Das Einstellen des Schwellenwerts benötigt mehrere iterative Schritte. Je höher der Schwellenwert liegt, desto weniger Features werden ausgewählt. Da es sich in diesem Fall bereits um sehr geringe Leistungsunterschiede handelt, wird ein sehr niedriger Schwellenwert mit 0.5 festgelegt. Die Transformation der Trainingsdaten und die Änderung im Testdatensatz erfolgt analog zu Kapitel 4.5.9.1.

Die Leistungsunterschiede der einzelnen Features werden in Abbildung 20 dargestellt. Die rote Linie zeigt den Schwellenwert des Transformators an. Alle Features, deren Leistungsunterschied unterhalb des Schwellenwertes liegen, werden aus dem Datensatz entfernt. In diesem Fall werden die Features „Marke“, „Ziel länge [mm]“, „Ziel_Breite“, „Ziel dicke [mm]“, „Anzahl gesamt Stiche“, „Dauer Gesamtstiche [s]“, „Temp“, „X“, „QuaVorsch“, „S“, „Vrg“, „ARBG“, „ArbPlatz“, „EinGew“, „ErzGew“, „aktAusStk“, „aktAusGew“, „Ist.Plan“, „SCHM“, „Rückmeld.“, „Material“, „EinDim1“, „EinDim3“, „ErzDim1“, „ErzDim2“, „ErzDim3“ und „DLZ“ aus dem Datensatz entfernt.

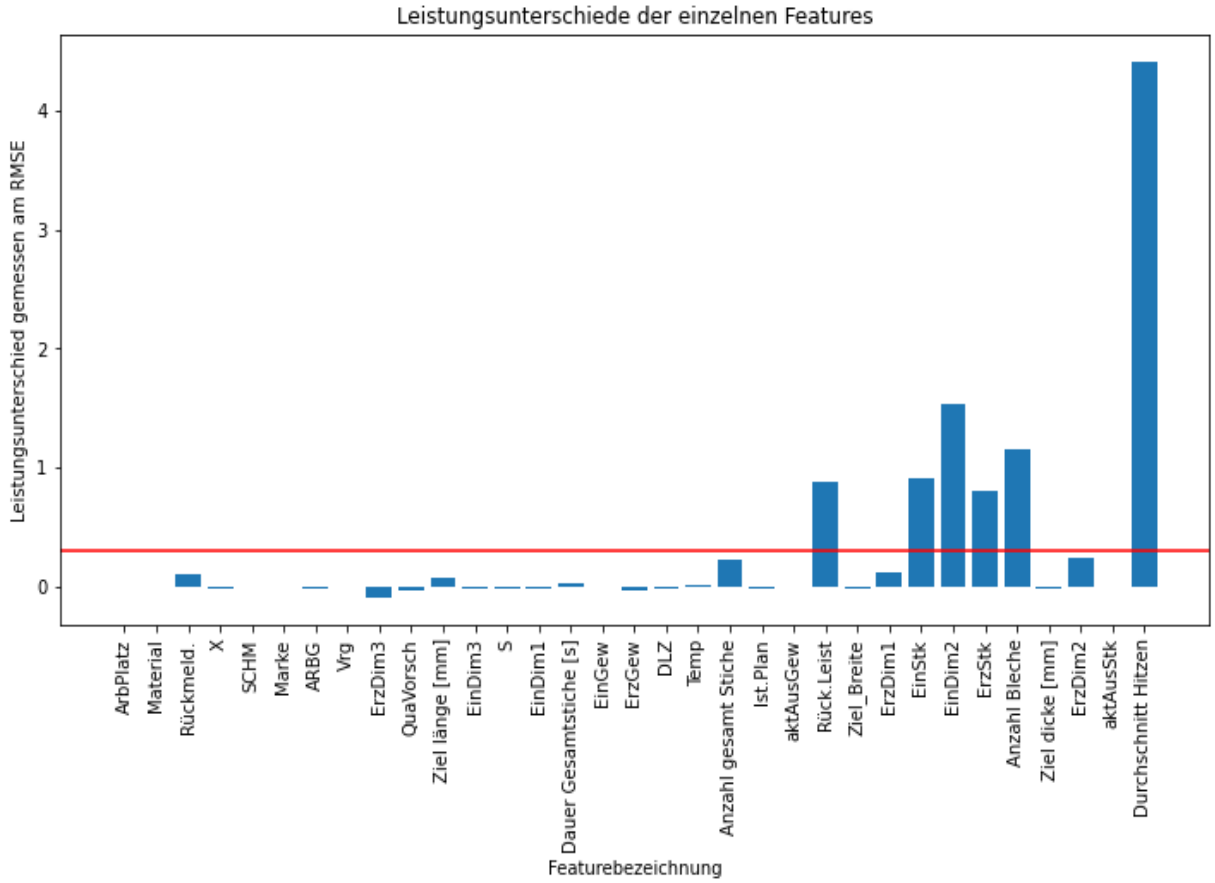

Abbildung 20: Leistungsunterschiede der einzelnen Features[121]

4.6 Modeling

Nachdem das Feature Engineering abgeschlossen ist, werden nun die SML- Modelle erstellt. Die Modelle werden mithilfe der Scikit- Learn Bibliothek von Python programmiert. Die Bibliothek ist unter dem unten angeführten Link aufrufbar.[122] In diesem Kapitel wird zunächst ein kurzer Einblick in die Scikit- Learn Bibliothek gegeben. Anschließend werden die ausgewählten Modelle vorgestellt und genauer beschrieben. Zuletzt wird die Implementierung in die Python-Oberfläche für die praktische Anwendung aufgezeigt.

4.6.1 Scikit- Learn

Die Bibliothek Scikit- Learn hat sich seit ihrer Veröffentlichung im Jahr 2007 zu einer der beliebtesten Open-Source-Bibliotheken für maschinelles Lernen in Python entwickelt. Die Bibliothek bietet Algorithmen für alle Aufgaben des maschinellen Lernens wie der Regression, der Klassifizierung aber auch Aufgaben aus dem unsupervised Machine Learning. Außerdem bietet Scikit- Learn Module für die Auswahl von Features, die Verarbeitung von Daten und die Auswertung von Modellen

[121] Eigene Darstellung
[122] https://scikit-learn.org/stable/index.html, letzter Zugriff: 26.01.2023

an. Scikit- Learn wurde als Erweiterung der *SciPy*-Bibliothek konzipiert und baut auf den Python-Bibliotheken *NumPy* und *matplotlib* auf. NumPy erweitert Python und unterstützt effizientere Operationen für mehrdimensionalen Matrizen. Matplotlib bietet Visualisierungswerkzeuge, etwa zur Darstellung von Ergebnissen, und SciPy bietet Module für wissenschaftliche Berechnungen. Scikit- Learn ist in der akademischen Forschung sehr beliebt, da es über eine gut dokumentierte, einfach zu verwendende und vielseitige API verfügt. Die Bibliothek kann ohne Einschränkungen in kommerziellen Anwendungen eingesetzt werden.[123]

4.6.2 Lineare Modelle

Lineare Modelle sind für viele Regressionsprobleme eine gute erste Wahl. Aufgrund ihres intuitiven Charakters können sie bereits sehr präzise Vorhersagen treffen. Eine gute Prognose von linearen Modellen setzt voraus, dass der Datensatz einen gewissen Grad an linearer oder polynomialer Zusammenhänge zwischen dem Eingangsvektor und dem Zielvektor aufweist. [124]

Für eine Regression werden gelabelte Datensätze vorausgesetzt, die einen Eingabevektor sowie einen Zielvektor beinhalten. Es handelt sich somit um einen überwachten Lernansatz. Die *einfache* lineare Regression stellt die einfachste Form der Regression dar, bei der versucht wird, eine gerade Linie an den Datensatz anzupassen. Die abhängige Variable (y) wird durch eine einzige unabhängige Variable (x) gesteuert.[125] Wenn der Eingangsvektor mehr als eine unabhängige Variable beinhaltet (x1, x2, x3 ...), wird die Regression als *Mehrfachregression* bezeichnet. Die Gleichung der Mehrfachregression wird in Gleichung 1 dargestellt.

$$y = a + b_1x_1 + b_2x_2 + b_3x_3 + \cdots b_nx_n + e$$

Gleichung 1: Formel für die Mehrfachregression[126]

Das Ziel der linearen Regression besteht darin, einen optimalen linearen Zusammenhang aller Eingangsvariablen mit der Zielvariable zu finden. Dazu werden die Koeffizienten (*a, b_1, …, b_n*) berechnet. Der Parameter e bildet den Fehlerterm der Gleichung.[127] Für den praktischen Anwendungsfall wird eine lineare Regression aus dem Paket *linear_model* importiert und definiert.[128]

[123] vgl. Hackeling, 2017, S. 16
[124] vgl. Johnston und Mathur, 2019, S. 111
[125] vgl. Ray, 2019, S. 36
[126] Tewari et al., 2020, S. 7
[127] vgl. Tewari et al., 2020, S. 7
[128] vgl. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression, (letzter Zugriff: 31.01.2023)

4.6.3 SVM – Support Vector Maschines

Support Vector Machines (SVM) können sowohl für Klassifizierungs- als auch Regressionsprobleme herangezogen werden. Für ein Klassifizierungsproblem muss zunächst eine Hyperebene definiert werden. Sie trennt die einzelnen Variablen in Klassen. Um diese Ebene zu beschreiben, wird die sogenannte *kernel- Funktion* herangezogen. Diese Funktion kann im einfachsten Fall linear sein. Sie kann aber auch sehr komplexe mathematische Funktionen beinhalten.[129]

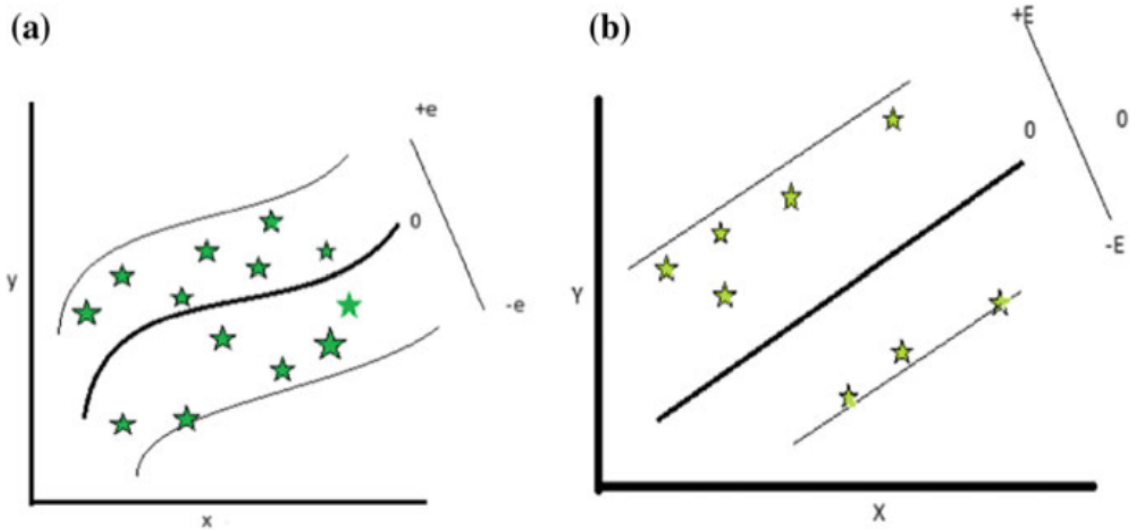

Abbildung 21: (a) nicht-lineare Ebene (b) lineare Ebene[130]

Wie bei der Klassifizierung zeichnet sich die Support-Vektor-Regression (SVR) ebenfalls durch die Verwendung dieser kernel-Funktionen aus. So hat sich die SVR als effektives Werkzeug für die Vorhersage von kontinuierlichen Werten erwiesen. Als überwachter Lernansatz verwendet die SVR eine symmetrische Verlustfunktion. Dabei wird eine e- Röhre, mit minimalem Radius *e,* symmetrisch um die geschätzte Funktion gebildet. Diese Funktion wird anschließend so minimiert, dass die flachste Röhre gefunden wird, welche die meisten Trainingsdaten enthält. Die Punkte mit dem kleinsten Abstand zur Hyperebene werden als Stützvektoren bezeichnet. Jene Fehlerwerte, die außerhalb dieses Schwellenwertes e liegen, werden von der Funktion ignoriert.[131] Abbildung 21 zeigt den Unterschied einer nicht-linearen und einer linearen Hyperbene mit ihrer Verlustfunktion. Eine genaue Beschreibung der komplexen Berechnungen zur SVR kann bei Awad et al. oder auf der Webseite von Scikit- Learn direkt nachgesehen werden.

[129] vgl. Ray, 2019, S. 37
[130] Tewari et al., 2020, S. 8
[131] vgl. Awad et al., 2015, S. 67

Die Vorteile des SVM liegen in seiner hohen Genauigkeit und der einfachen Auswahl der kernel-Funktionen. Zudem ist seine Performance unabhängig von der Anzahl der Features. Als Nachteil ist die schwierige Verständlichkeit der komplexen Algorithmen zu nennen. Weiters ist die Leistung des Algorithmus von der Auswahl der Features abhängig.[132]

Scikit- Learn bietet drei verschiedene Implementierungsformen der Support-Vektor-Regression an: *SVR, NuSVR* und *LinearSVR*. LinearSVR bietet eine schnellere Implementierung als SVR, berücksichtigt aber nur den linearen Kernel, während NuSVR eine etwas andere Formulierung als SVR und LinearSVR implementiert.[133] Für diese Arbeit wird die gewöhnliche SVR verwendet. Die SVR wird aus der Bibliothek *svm* aufgerufen.[134]

4.6.4 Regressionsbaummodelle

4.6.4.1 Entscheidungsbaum

Der Entscheidungsbaum (im Englischen: Decision Tree) ist ein beliebter Ansatz des überwachten maschinellen Lernens zur Lösung von Klassifizierungs- und Regressionsproblemen durch kontinuierliche Aufteilung von Daten auf der Grundlage eines gewählten Parameters.[135]

Das Ziel des Algorithmus besteht darin, ein Modell anzulernen, das die Zielwerte auf der Grundlage einfacher Entscheidungsregeln (zum Beispiel: Wenn-Dann-Bedingungen) vorhersagen kann. Der Entscheidungsbaum beginnt an der Wurzel des Baumes und unterteilt den Datensatz in zwei oder mehr sich nicht überschneidende Teilmengen, die jeweils als Knoten (*nodes*) der Wurzel dargestellt werden. Anhand eines bestimmten Parameters wird diese Aufteilung an jedem Knoten fortgeführt, bis der letzte Knoten (*leaf node*) erreicht wird, der den Zielwert beinhaltet.[136] Bei Klassifizierungsaufgaben stellen die letzten Knoten des Entscheidungsbaums Klassen dar. Bei Regressionsaufgaben können die Werte der Zielvariablen gemittelt werden, um den Schätzwert für die Antwortvariable zu ermitteln. Nachdem der Entscheidungsbaum mithilfe der Trainingsdaten konstruiert wurde, braucht man für eine Vorhersage für eine Testinstanz nur den Kanten zu folgen, bis ein Blattknoten erreicht wird.[137]

[132] vgl. Singh et al., 2016
[133] vgl. https://scikit-learn.org/stable/modules/svm.html#regression, (letzter Zugriff: 31.01.2023)
[134] vgl. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR, (letzter Zugriff: 31.01.2023)
[135] vgl. Ray, 2019, S. 37
[136] vgl. Sarkar et al., 2018, S. 323 ff
[137] vgl. Hackeling, 2017, S. 98

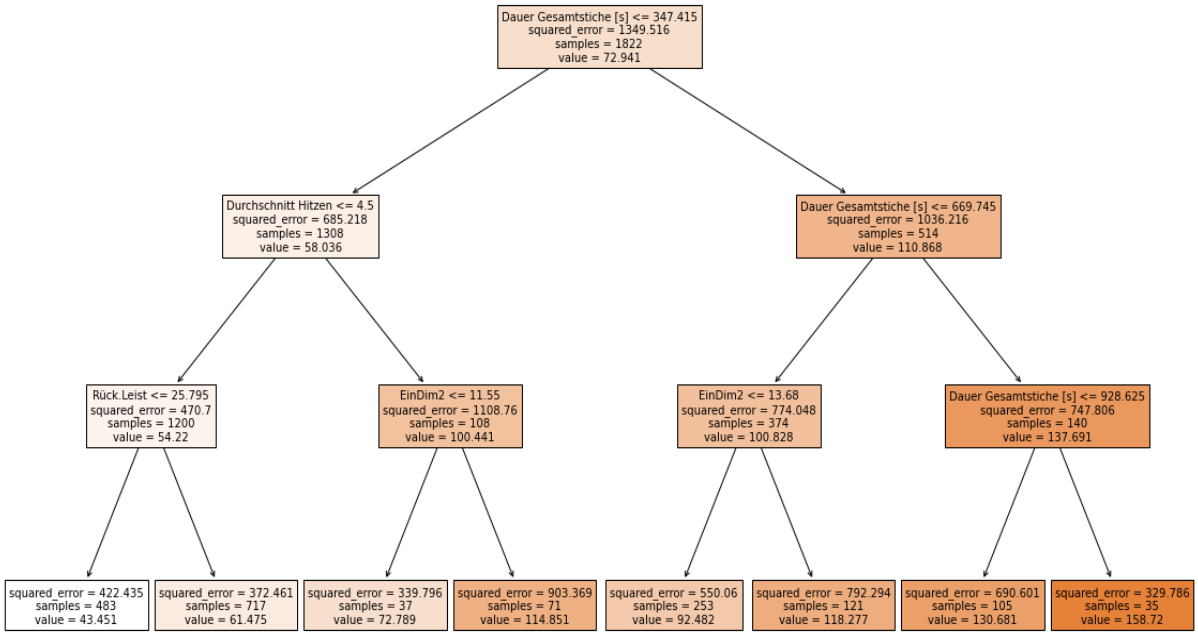

Abbildung 22: Dreistufiger Regressionsbaum[138]

Abbildung 22 zeigt einen Regressionsbaum mit drei Knoten, der im Rahmen der Modellierung für den verwendeten Datensatz ermittelt wurde. Dafür wurde der *Decision Tree Regressor* verwendet und nach dem Handbuch antrainiert.[139] Die Features werden anhand ihrer Wichtigkeit für das Modell, ähnlich der rekursiven Feature- Auswahl aus Kapitel 4.5.9.2, durchgeführt. Die Messung der Leistung wird mithilfe des MSE gemessen.[140]

4.6.4.2 Random Forest

Der Random Forest (zu deutsch: Zufallswald) ist eine Erweiterung des einfachen Entscheidungsbaums. Die Idee des Random Forest Algorithmus besteht darin, die Trainingsmenge zunächst in Teilmengen zu unterteilen. Für jede dieser Teilmengen wird anschließend ein eigener Entscheidungsbaum konstruiert. Dieser Vorgang geschieht nach dem Zufallsprinzip, weshalb der Random Forest auch seinen Namen erhalten hat. Der Random Forest ist die Zusammenfassung aller dieser Entscheidungsbäume. Er kann sowohl für Klassifikations- als auch Regressionsprobleme herangezogen werden.[141] Die oben beschriebene Methode, welcher der Random Forest zuzuschreiben ist, wird in der Literatur auch als *Ensemble*

[138] Eigene Darstellung
[139] vgl. https://scikit-learn.org/stable/modules/tree.html#regression, (letzter Zugriff: 31.01.2023)
[140] vgl. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor.feature_importances_, (letzter Zugriff: 31.01.2023)
[141] vgl. Jo, 2021, S. 160f

Methode bezeichnet. Die Vorteile des Random Forest bestehen in seiner schnellen Skalierbarkeit, seiner Robustheit gegenüber Ausreißern und einer möglichen Visualisierung der Ergebnisse. Mit zunehmender Anzahl der Entscheidungsbäume nimmt allerdings auch die Berechnungsdauer des Algorithmus zu.[142]

Das Ensemble- Modul bietet mehrere Random Forest-Algorithmen, wie den in dieser Arbeit verwendeten *Random Forest Regressor*. Der Zweck des Algorithmus besteht darin, die Varianz des Modells zu verringern. Einzelne Entscheidungsbäume weisen in der Regel eine hohe Varianz auf und neigen zu einer Überanpassung des Modells. In der Praxis ist die Varianzreduzierung oft signifikant, was zu einem insgesamt besseren Modell führt. Die Vorhersage des Ensembles ergibt sich aus der gemittelten Vorhersage der einzelnen Regressoren.[143]

4.6.4.3 Gradient Tree Boosting

Ein weiterer Algorithmus aus der Familie der Ensemble Methoden, ist der *Gradient Tree Boosting*. *Gradient Boosting* ist eine Familie von Algorithmen, die schwache ML-Algorithmen in Starke umwandeln sollen. Boosting-Algorithmen beginnen mit dem Training eines Basis-Algorithmus und passen dann die Verteilung der Trainingsproben entsprechend dem Ergebnis des Basis- Algorithmus so an, dass falsche Ergebnisse von nachfolgenden Basis-Algorithmen mehr Aufmerksamkeit erhalten. Nach dem Training des ersten Basis- Algorithmus wird der zweite Basis- Algorithmus mit den angepassten Trainingsstichproben trainiert, und das Ergebnis wird verwendet, um die Verteilung der Trainingsstichproben erneut anzupassen. Dieser Prozess wird so lange wiederholt, bis die Anzahl der Algorithmen einen vordefinierten Wert erreicht hat. Die einzelnen Basis- Algorithmen werden schließlich gewichtet und miteinander kombiniert.[144]

In dieser Arbeit wird ein Gradient Boosting Tree Regressor (GBR) aus dem Ensemble-Modul verwendet. Die Funktion des Algorithmus ist an die Arbeit von Friedman aus dem Jahr 2015 angelehnt.[145]

4.6.5 Artificial Neural Network

Das *(künstliche) neuronale Netzwerk* ist heute ein breites und interdisziplinäres Forschungsgebiet mit vielen unterschiedlichen Definitionen. Die Grundidee liegt darin, dass neuronale Netzwerke in der gleichen Art und Weise mit der Welt interagieren sollen, wie es das biologische Nervensystem es tut. Aus der Sicht der Informatik

[142] vgl. Singh et al., 2016
[143] vgl. https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees, (letzter Zugriff: 31.01.2023)
[144] vgl. Zhou, 2021, S. 184
[145] vgl. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html#sklearn.ensemble.GradientBoostingRegressor, (letzter Zugriff: 04.02.2023)

können wir ein neuronales Netz als ein mathematisches Modell mit vielen Parametern und verschachtelten Funktionen betrachten, das auf der Grundlage des Nervensystems basiert.[146] Das kleinste Element des neuronalen Netzwerks bildet das Neuron, das mit anderen Neuronen regelmäßig im Austausch steht. Bei maschinellen Lernverfahren wird ein einzelnes Neuron als Recheneinheit modelliert. Es enthält mehrere Werte als Eingang und sendet einen einzigen Wert als Ausgabe weiter. Der endgültige Ausgabewert wird durch die Anwendung einer Aktivierungsfunktion auf die Eingabewerte berechnet. Das künstliche Neuron wird als eine Art „Blackbox" betrachtet, die aus mehreren Eingabewerten einen einzigen Ausgabewert errechnet.[147]

Das mehrschichtige Perzeptron (MLP) ist eines der am häufigsten verwendeten künstlichen neuronalen Netze. Das MLP besteht aus mehreren Schichten künstlicher Neuronen. Die Schichten des MLP bilden einen gerichteten und azyklischen Graphen. Im Allgemeinen ist jede Schicht vollständig mit der nachfolgenden Schicht verbunden. Die Ausgabe jedes künstlichen Neurons in einer Schicht ist eine Eingabe für jedes künstliche Neuron in der nächsten Schicht in Richtung der Ausgabe. MLPs haben in der Regel drei oder mehr Schichten von künstlichen Neuronen.[148]

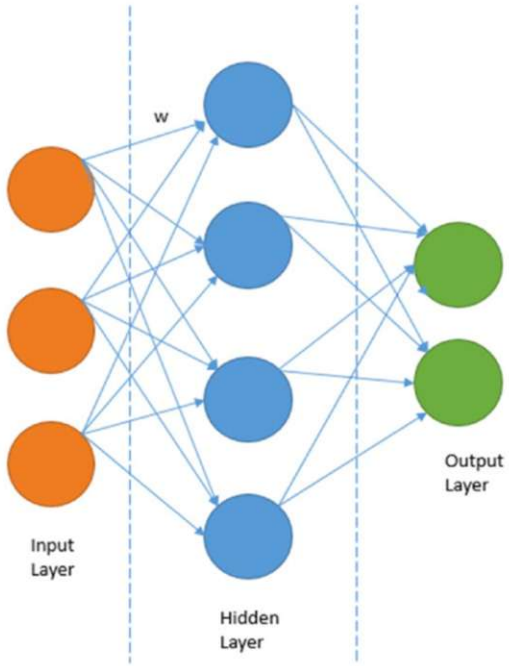

Abbildung 23: Einfaches neuronales Netzwerk[149]

[146] vgl. Zhou, 2021, S. 104
[147] vgl. Jo, 2021, S. 18
[148] vgl. Hackeling, 2017, S. 189f
[149] Sarkar et al., 2018, S. 103

Abbildung 23 zeigt den Aufbau eines einfachen neuronalen Netzwerkes mit drei Schichten. Das MLP kann neben der Eingangschicht (Input Layer) eine oder mehrere versteckte Schichten (Hidden Layer) aufweisen. Die Ausgabeschicht (Output Layer) präsentiert schließlich den Zielvektor.[150]

Der MLP-Regressor implementiert ein MLP für Regressionsprobleme in Python. Der MLP-Regressor verwendet die Verlustfunktion des mittleren quadratischen Fehlers. Genauere Berechnungsschritte können der Webseite entnommen werden.[151] Für die praktische Anwendung wird der MLP-Regressor aus dem Paket *neural_network* importiert und definiert.[152]

4.6.6 Modellierung in Python mit Scikit- Learn

Um die oben beschriebenen SML-Modelle in Python implementieren zu können, müssen sie zunächst aus der Bibliothek sklearn importiert und definiert werden. Für die Definition der Algorithmen werden keine zusätzlichen Parameter eingestellt. Anschließend werden die SML-Modelle für den Datensatz mithilfe der Fit - Methode trainiert. Als Eingabewerte werden den Algorithmen die Trainingsdaten in Form des Eingabevektors und des Zielvektors übergeben. Nachdem die Modelle durch die Trainingsdaten moduliert wurden, werden für die Evaluierung ihrer Leistung Vorhersagen getroffen. Dies geschieht in Python mit der Predict- Methode. Als Eingabevektor erhält die Funktion nun den Eingabevektor des Testdatensatzes.[153] Der Quellcode zur Modellierung der Algorithmen ist dem Anhang 7.2 zu entnehmen.

4.7 Evaluierung

In der letzten Phase der Evaluation (zu deutsch: Evaluierung) werden die zuvor ausgewählten SML-Modelle hinsichtlich ihrer Vorhersagegenauigkeit überprüft und miteinander verglichen. Das Ziel besteht schließlich darin, das beste Modell für das vorliegende Problem auszuwählen.

Ein SML-Regressionsmodell liefert für eine Eingangsvariable X eine vorhergesagte Zielgröße y als kontinuierlichen Wert. Im Idealfall kann das Modell den Wert exakt vorhersagen und der vorhergesagte Wert entspricht dem tatsächlichen Wert. In der Praxis besteht zwischen der Modellvorhersage und den tatsächlichen Werten eine Differenz. Je geringer diese Differenz ausfällt, desto besser ist die Leistung des Modells. Um diese Differenz zu messen, kann jedoch nicht einfach der Mittelwert für

[150] vgl. Gardner und Dorling, 1998, S. 2627f
[151] vgl. https://scikit-learn.org/stable/modules/neural_networks_supervised.html#regression, (letzter Zugriff: 02.02.2023)
[152] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor, (letzter Zugriff: 02.02.2023)
[153] https://scikit-learn.org/stable/developers/develop.html, (letzter Zugriff: 03.03.2023)

die Abweichungen über alle Datenpunkte herangezogen werden, da es Datenpunkte geben könnte, die einen positiven oder negativen Vorhersagefehler aufweisen. Bei der Aufsummierung würden viele der Fehler ausgeglichen und die Leistung des Modells verfälscht gemessen werden. Aus diesem Grund werden eigene Fehlerfunktionen zur Messung der Modellleistung herangezogen.[154]

4.7.1 Methode zur Ermittlung der Modelleistung

Um die die Leistung der antrainierten SML-Algorithmen vergleichen zu können sollen Kennzahlen herangezogen werden. Dazu wird in einem ersten Schritt die Genauigkeit der Modelle nach Lingitz et al. mit den bereits in Kapitel 3.2 genannten verschiedenen Fehlermaßstäben gemessen.[155] Im Folgenden werden diese Fehler genauer beschrieben:

Der **MAE** ist eine lineare Bewertungsfunktion. Die Funktion teilt jeder Differenz die gleiche Gewichtung zu, während die Fehler zusammenfasst werden. Der MAE kann jeden Wert zwischen Null und Unendlich annehmen und ist unabhängig vom Vorzeichen der Fehler. Damit das Vorzeichen des Fehlers die Leistungsschätzung nicht beeinflusst, kann auch das Quadrat der Fehlerterme herangezogen werden. Nimmt man den Mittelwert der quadrierten Fehler, erhält man den mittleren quadratischen Fehler **MSE.**

Während der MAE die gleiche Einheit wie die Zielvariable y hat, ist die Einheit des MSE die quadrierte Einheit von y. Das macht den MSE bei der Beurteilung des Modells in der Praxis etwas weniger interpretierbar. Wenn die Quadratwurzel aus dem MSE gezogen wird, erhält man den **RMSE**. Er behält wieder die gleiche Einheit, wie die Zielvariable und ist daher besser interpretierbarer. Da die Fehler quadriert werden, bevor sie gemittelt werden, können selbst einige wenige hohe Fehlerwerte den RMSE-Wert erheblich ansteigen lassen. Das bedeutet, dass der RMSE-Wert für die Beurteilung von Modellen, bei denen große Fehler bestraft werden sollen nützlicher ist als der MAE-Wert.[156]

Der **MAPE** wird berechnet, indem der absolute Fehler für jede Beobachtung durch den beobachteten Wert geteilt wird. Die resultierenden Prozentsätze werden anschließend gemittelt. Der MAPE ist dann nützlich, wenn die Größe oder der Umfang einer Vorhersagevariablen für die Bewertung der Genauigkeit der Vorhersage von Bedeutung ist. Er gibt an, wie groß der Fehler bei der Vorhersage im Vergleich zum

[154] vgl. Johnston und Mathur, 2019, S. 270
[155] vgl. Lingitz et al., 2018, S. 4
[156] vgl. Johnston und Mathur, 2019, S. 270

tatsächlichen Wert ist.[157] Der **NRMSE** gibt den durchschnittlichen Vorhersagefehler in Prozent des tatsächlichen Wertes an.[158]

Abkürzung	Bezeichnung	Formel
MAE	Mittlerer absoluter Fehler	$\frac{1}{n}\sum_{i=1}^{N}\lvert P_i - R_i\rvert$
MAPE	Mittlerer absoluter prozentualer Fehler	$\frac{100}{n}\sum_{i=1}^{N}\frac{\lvert P_i - R_i\rvert}{R_i}$
MSE	Mittlerer quadratischer Fehler	$\frac{1}{n}\sum_{i=1}^{N}(P_i - R_i)^2$
RMSE	Wurzel des mittleren quadratischen Fehlers	$\sqrt[2]{\frac{1}{n}\sum_{i=1}^{N}(P_i - R_i)^2}$
NRMSE	Normalisierten mittlerer quadratischer Fehler	$100 * \frac{\sqrt[2]{\frac{1}{n}\sum_{i=1}^{N}(P_i - R_i)^2}}{R_{max} - R_{min}}$

Tabelle 6: Berechnung der Evaluierungskennzahlen[159]

Tabelle 6 zeigt die Formeln, die für die Berechnung der Evaluierungskennzahlen herangezogen werden. P_i gibt den mithilfe des Modells vorausgesagten Zielwert an und R_i den tatsächlichen Zielwert.

Für die Implementierung in Python wird das Paket *metrics* aus Skit-learn herangezogen. Der NRMSE ist in diesem Paket nicht enthalten und wird deshalb manuell programmiert. Die Eingabewerte stellen die tatsächlichen sowie die prognostizierten Zielgrößen der einzelnen SML-Algorithmen dar.[160] (siehe Anhang 7.3)

4.7.2 Visualisierung der Planungsqualität

Die Ergebnisse aus der Literaturrecherche zur Planungsqualität aus Kapitel 3 haben gezeigt, dass es bis heute keine einheitliche Definition zur Planungsqualität gibt. Alle Definitionen haben jedoch gemeinsam, dass eine gute Planungsqualität im maschinellen Lernen eine geringe Abweichung des geplanten Ereignisses mit dem

[157] vgl. Khair et al., 2017, S. 2
[158] vgl. Lingitz et al., 2018, S. 4
[159] vgl. Johnston und Mathur, 2019, S. 271; vgl. Khair et al., 2017, S. 2; vgl. Lingitz et al., 2018, S. 4
[160] https://scikit-learn.org/stable/modules/model_evaluation.html#model-evaluation, (letzter Zugriff: 03.02.2023)

tatsächlichen Ereignis beinhaltet. Um die SML- Modelle hinsichtlich ihrer Planungsqualität visuell vergleichen zu können, werden die Formeln zur Berechnung der Standardabweichung herangezogen. Dafür wird zunächst die Differenz der Abweichungen berechnet und anschließend die Abweichungen in einem Häufigkeitsdiagramm dargestellt. Darüber wird die Verteilungsfunktion der Normalverteilung gelegt, welche die Verteilung der Häufigkeit abbildet. Der Mittelwert μ entspricht dem Mittelwert der Abweichungen. Die Standardabweichung σ ist mit dem RMSE gleichzusetzen. Die Formeln, die zur Visualisierung der Planungsqualität herangezogen werden, sind in Tabelle 7 aufgelistet.

Abkürzung	Bezeichnung	Formel
μ	Mittelwert	$\frac{1}{n}\sum_{i=1}^{N} P_i - R_i$
σ	Standardabweichung	$\sqrt[2]{\frac{1}{n}\sum_{i=1}^{N}(P_i - R_i)^2}$

Tabelle 7: Formeln zur Visualisierung der Planungsqualität

5 Ergebnisse und Auswertung

Die Erstellung der SML-Modelle wurde nach dem Cross Industry Standard Process for Data Mining (CRISP-DM) Prinzip vorgenommen. Dafür wurden die Schritte *Data Preparation, Modelling* und *Evaluation* iterativ durchlaufen. Nach jedem Durchlauf wurden in der Phase der Evaluierung die Kennzahlen ermittelt, um das beste Modell zur Prognose der Zielgröße zu identifizieren. Der Durchlauf wurde wiederholt, bis sich gegenüber dem Referenzmodell (siehe Kapitel 4.3) zufriedenstellende Ergebnisse eingestellt haben. Als Ziel wurde eine Reduzierung der Abweichung um 10 % vordefiniert. Dieses Ziel wurde angestrebt. Schließlich wird jenes Modell ausgewählt, welches in der Evaluierungsphase die beste Leistung erzielt. Informationen zum CRISP-DM Prinzip können in Kapitel 3.1 nachgelesen werden. Bei der Auflistung der Ergebnisse muss beachtet werden, dass die Ergebnisse für jede Neuberechnung minimale Unterschiede aufweisen.

5.1 Ergebnisse der Feature Auswahl

Im Rahmen der Modellerstellung wird, nach der Datenvorbereitung durch Feature Engineering, die Feature Auswahl nach Kapitel 4.5.9 vorgenommen. Dabei werden die zwei Methoden Auswahl der Features nach ihrer Einzelleistung (FAE) und Rekursive Feature Eliminierung (RFE) aus dem Paket Feature Engine getestet. Tabelle 8 repräsentiert die Ergebnisse aus der Phase der Evaluierung für jeden Iterationsschritt. Zunächst wurden die Modelle ohne Feature Auswahl getestet. Anschließend wurden die beiden oben genannten Methoden implementiert und die Evaluierung erneut durchgeführt. Als Fehlerfunktion wird der RMSE herangezogen.

Feature Auswahl (FA)	Anzahl der Features	RT	LM	SVR	ANN	RF	GBR
Ohne FA	33	25,94	24,11	36,25	22996,18	17,65	16,81
FAE	11	25,74	22,06	30,16	20,64	18,62	17,34
RFE	6	23,52	20,45	24,98	18,79	19,13	18,44

Tabelle 8: Vergleich der Feature Auswahl Methoden anhand des RMSE[161]

Die Ergebnisse zeigen, dass beide Methoden zur Feature Auswahl Verbesserungen der Modellleistungen für die Support Vektor Regression (SVR), sowie das neuronale Netzwerk (ANN) und das lineare Regressionsmodell (LM) erzielen konnten. Der hohe Wert der ANN für den ersten Iterationsschritt rührt daher, dass die Funktion durch die hohe Anzahl von 33 Features nicht konvergieren konnte. Mit sinkender Anzahl der

[161] Eigene Darstellung

Features kann der Algorithmus in seiner Optimierung konvergieren und liefert sehr gute Ergebnisse. Für die übrigen Algorithmen, den Regressionsbaum (RT), die Random Forest Regression (RF) und die Gradient Boosting Maschine (GBR), konnten durch die Methoden zur Feature Auswahl keine signifikanten Änderungen in ihrer Leistung wahrgenommen werden. Weniger Features bedeuten allerdings, dass weniger Daten zur Ermittlung der Zielgrößen herangezogen werden müssen. Die rekursive Feature Eliminierung konnte in diesem Durchlauf den Datensatz auf 6 Features reduzieren. Dadurch wird nicht nur die Berechnungszeit verkürzt, sondern die Plausibilität der Ergebnisse eindeutiger. Die genauen Evaluierungsergebnisse aller SML-Methoden für die jeweilige Feature Auswahl Methode können dem Anhang 7.4 entnommen werden.

5.2 Vergleich der SML-Algorithmen

Um die erstellten SML-Modelle hinsichtlich ihrer Planungszuverlässigkeit miteinander vergleichen zu können, werden die in Kapitel 4.7 beschriebenen Fehlerfunktionen herangezogen. Die Ergebnisse sind der Tabelle 9 zu entnehmen. Für die Tabelle wurden die jeweils besten Ergebnisse für jeden SML-Algorithmus ausgewählt.

Fehlerfunktion	**RT**	**LM**	**SVR**	**ANN**	**RF**	**GBR**
Feature Auswahl (FA)	RFA	Ohne FA	RFA	RFA	Ohne FA	Ohne FA
MAE	16,44	14,33	18,03	14,37	13,56	12,71
MAPE	28,46	25,49	31,10	25,75	23,99	22,85
MSE	553,39	581,30	623,80	353,03	335,32	282,53
RMSE	23,52	24,11	24,98	18,79	18,31	16,81
NRMSE	13,44	13,78	14,27	10,74	10,64	9,60

Tabelle 9: Vergleich der SML-Modelle mithilfe der Fehlerfunktionen[162]

Die Analyse zeigt, dass alle verwendeten SML-Algorithmen sehr ähnliche Ergebnisse für die Vorhersage der Prozesszeiten für den konkreten Anwendungsfall liefern. Das beste Ergebnis konnte mit einem GBR-Algorithmus werden, der auf dem Entscheidungsbaummodell aufbaut. Dabei wurde kein Feature Auswahl Algorithmus verwendet und 33 Features zur Ermittlung der Zielgröße herangezogen. Unter den konventionellen SML-Modellen konnte das neuronale Netzwerk die besten Ergebnisse liefern. Die Ergebnisse konnten mithilfe der rekursiven Feature Eliminierung und 6 Features erzielt werden. Das lineare Modell, welches das einfachste und verständlichste Modell aller getesteten Modelle darstellt, liefert ebenfalls sehr gute Ergebnisse.

[162] Eigene Darstellung

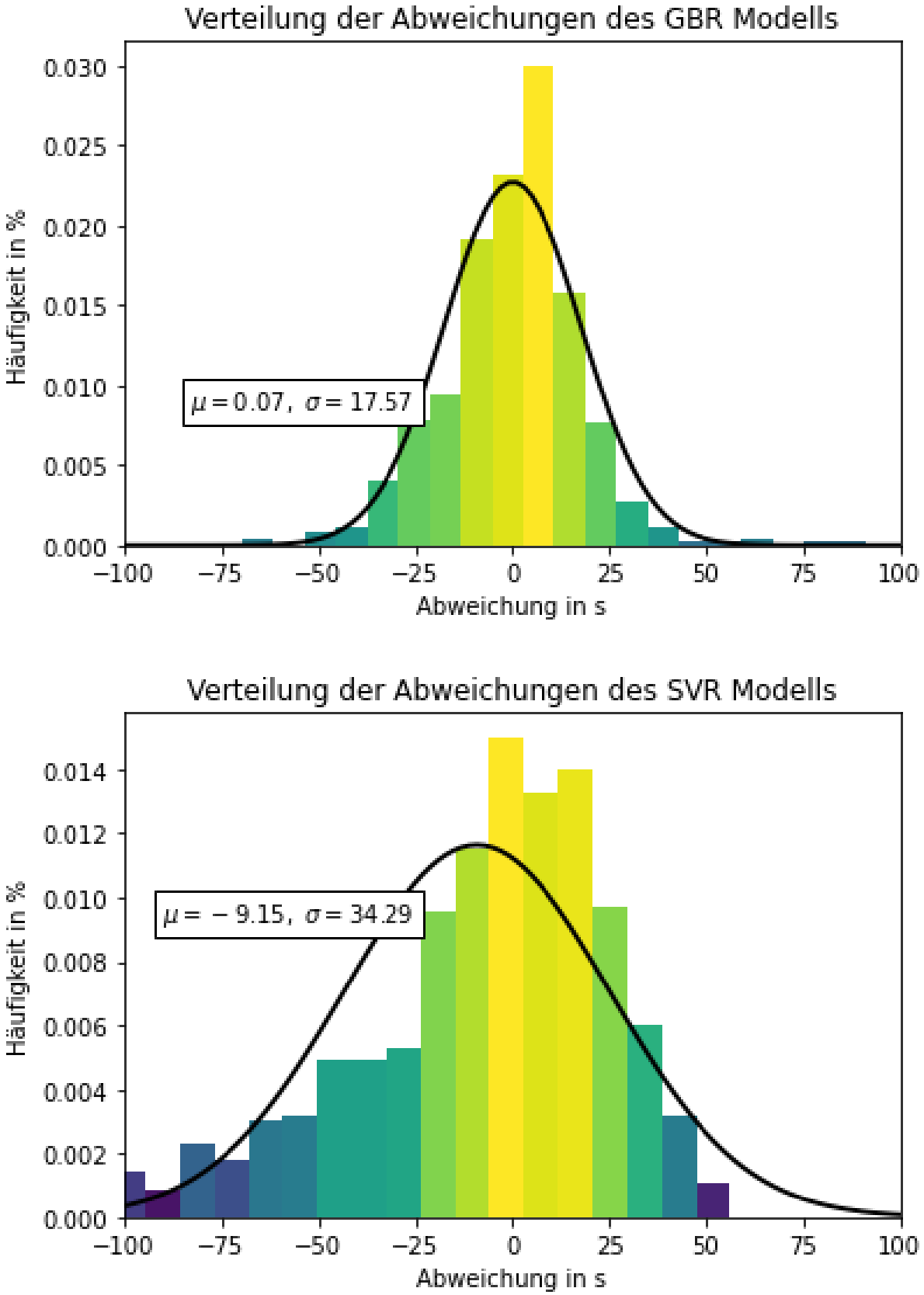

Abbildung 24: Vergleich des GBR-Modells mit dem SVR-Modell hinsichtlich der Planungsqualität[163]

[163] Eigene Darstellung

Schließlich erfolgt die visuelle Darstellung der Planungsqualität nach Kapitel 4.7.2. Dazu wird in Abbildung 24 beispielhaft die Planungsqualität des besten Modells (GBR-Modell) mit der Planungsqualität des SVR- Modells verglichen, welches im konkreten Fall die niedrigste Leistung erzielen konnte. Für beide Modelle wurden 33 Features zur Ermittlung der Leistung herangezogen. Zu sehen sind die deutlich geringeren Abweichungen und die daraus resultierende bessere Planungsqualität des LM-Modells. Die visualisierte Planungsqualität für jedes einzelne SML-Modell kann dem Anhang 7.5 entnommen werden. Die visualisierte Planungsqualität wurde für die Ergebnisse aus Tabelle 9 ermittelt.

5.3 Ergebnisse in Bezug auf den Referenzansatz

Als Ziel dieser Arbeit wurde eine Reduktion der Abweichungen von mindestens 10 Prozent gegenüber den Abweichungen des bereits entwickelten Vorhersagemodells gesetzt. Die Leistung des Referenzansatzes wurde zunächst mithilfe der Wurzel des mittleren quadratischen Fehlers gemessen. Um einen besseren Vergleich durchzuführen, wurde der Referenzansatz um die übrigen Fehlerfunktionen erweitert. Weitere Informationen zum Referenzansatz finden sich in Kapitel 4.3.

Fehlerfunktion	GBR	Referenzansatz	Fehleränderung
MAE	12,71	21,09	39 %
MAPE	22,85	27,13	15 %
MSE	282,53	5.662,25	95 %
RMSE	16,81	75,25	77 %
NRMSE	9,60	4,77	-101 %

Tabelle 10: Fehleränderung in Bezug auf den Referenzansatz[164]

Tabelle 10 zeigt die Ergebnisse des Gradient Boosting Regression (GBR)- Algorithmus im Vergleich zu den Ergebnissen des Referenzansatzes. Es zeigt sich, dass der gewählte Algorithmus im Vergleich zum Referenzsatz deutlich bessere Ergebnisse erzielt. Bezogen auf die den RMSE konnte eine Verbesserung der Leistung um etwa 77 Prozent erzielt werden. Betrachtet man den MAPE, so konnte eine Verbesserung der Leistung um 15 Prozent erzielt werden. Die Berechnung der Fehleränderung erfolgt aus dem Quotienten der Fehlerdifferenz zum Fehlerwert des Referenzansatzes. Weitere Werte sind der Tabelle 10 zu entnehmen.

[164] Eigene Darstellung

6 Zusammenfassung und Ausblick

Die Vorhersage von Zielgrößen in der Industrie wird für die PPS mittels konventioneller Vorhersagemodelle immer schwieriger. Schuld daran ist unter anderem die zunehmende Komplexität in der Produktion. Eine genaue Vorhersage der Produktionsfaktoren ist jedoch wichtig, um unnötige Verschwendungen zu vermeiden und Liefertermine einhalten zu können. Maschinelle Lernalgorithmen weisen in der Literatur großes Potential auf, um die Zuverlässigkeit der Vorhersagen in der Industrie zu verbessern. Im Rahmen dieser Arbeit wurde deshalb der übergeordneten Forschungsfrage nachgegangen, *wie unter der Anwendung von Supervised Machine Learning (SML) die Prognosegenauigkeit von Zielgrößen in der PPS verbessert werden kann.* Im Folgenden Kapitel werden zunächst mithilfe der Ergebnisse der Arbeit die Forschungsfragen beantwortet. Anschließend werden mögliche Schritte zur Weiterentwicklung der Arbeit aufgezeigt.

6.1 Reflexion der Forschungsfragen

F1) Wie kann unter der Anwendung von SML die Prognosegenauigkeit von Zielgrößen in der PPS verbessert werden?

Die Anwendung von SML – Algorithmen in der PPS erfolgte in dieser Arbeit in drei Hauptschritten: In einem ersten Schritt wurden zunächst einige Studien herangezogen, die den Einsatz von SML in der PPS untersucht haben. Die Erkenntnisse daraus wurden genutzt, um relevante Algorithmen anhand ihrer Häufigkeit der Verwendung in der Literatur auszuwählen und hinsichtlich ihrer Leistung zu testen. Um die Leistung der einzelnen Algorithmen zu messen, wurden die Algorithmen auf ein reales Fallbeispiel aus der Stahlindustrie angewendet. Dazu wurde der Datensatz in einem zweiten Schritt mithilfe von Feature Engineering vorbereitet. Dieser Schritt stellt im Rahmen der Modellerstellung den aufwändigsten, aber auch wichtigsten Schritt dar. In einem letzten Schritt wurden die SML- Algorithmen modelliert und hinsichtlich ihrer Prognosegenauigkeit getestet. Die Evaluierung erfolgte dabei durch fünf Fehlerfunktionen. Zum Vergleich der Ergebnisse wurde zusätzlich ein zuvor entwickelter Referenzansatz herangezogen, der ohne Feature Engineering entwickelt wurde. Zuletzt wurde schließlich die Definition der Planungsqualität in der Literatur geklärt. Die Planungsqualität der SML-Algorithmen wurde mithilfe der Formeln für die Standardabweichung visualisiert. Die zugrundeliegende Methode zur Erstellung der Modelle erfolgte mit dem Cross Industry Standard Process for Data Mining (CRISP-DM).

F 1.1) Welche SML-Methoden zur Prognose von Zielgrößen in der PPS werden in der Literatur beschrieben?

Die Beantwortung dieser Frage erfolgte im Rahmen der Recherche zum aktuellen Stand der Technik in Kapitel 3.2. Dazu wurden einige relevante Lösungsansätze zur Anwendung von SML in der PPS unter Zuhilfenahme von Feature Engineering untersucht. Die Ergebnisse der Analyse zeigen, dass zur Vorhersage von Zielgrößen in der Industrie viele verschiedene SML-Algorithmen verwendet werden. Häufig werden mehrere SML- Algorithmen gleichzeitig getestet. Zu den wichtigsten Algorithmen zählen Lineare Modelle, neuronale Netzwerke, Support Vector Modelle und Regressionsbaummodelle unterschiedlicher Arten. Alle SML-Modelle wurden mithilfe von Feature Engineering vorbereitet. Zudem wurden in den ausgewählten Studien ausschließlich Regressionsmodelle zur Vorhersage der Zielgrößen verwendet. Der Grund dahinter ist, dass Zielgrößen in der Industrie in den meisten Fällen kontinuierliche Werte darstellen.

F1.2) Wie hoch ist die Zuverlässigkeit ausgewählter SML- Algorithmen für die Prognose von Zielgrößen in der PPS, speziell in der Stahlindustrie?

Die Frage nach der Zuverlässigkeit wurde im Verlauf dieser Arbeit zunächst im Rahmen der Literaturrecherche in Kapitel 3.2. geklärt. Es zeigt sich, dass besonders Entscheidungsbaummodelle sehr gute Prognosen in der PPS liefern können. Der einfache Entscheidungsbaum (RT) hat dabei weniger überzeugt als die Ensemble Methoden, welche die Kombination mehrerer Entscheidungsbäume nutzen. Dazu zählen das Random Forrest Modell (RF) sowie die Gradient Tree Boosting Maschine (GBR). In allen Studien konnten einfache lineare Modelle (LM) ebenfalls zufriedenstellende Ergebnisse liefern. Diese haben sich hinsichtlich ihrer Einfachen Implementierung und Nachvollziehbarkeit bewährt.

In Kapitel 4 wurde schließlich die Prognosegenauigkeit für ausgewählte SML-Algorithmen speziell in der Stahlindustrie untersucht. Dabei zeigten sich ähnliche Ergebnisse wie aus den Erkenntnissen aus Kapitel 3. Unter den gängigen SML-Algorithmen konnte ein Neuronales Netzwerk die beste Vorhersagegenauigkeit erzielen. Dafür musste allerdings ein Feature Auswahl-Algorithmus herangezogen werden. Das beste Ergebnis in der Analyse lieferte die Ensemble Methode GBR, ohne Feature Auswahl-Methode.

Zusammenfassend lässt sich sagen, dass alle der ausgewählten Algorithmen durch Zuhilfenahme des Feature Engineering sehr gute Ergebnisse liefern konnten. Während die Vorhersagen von Zielgrößen durch traditionelle Modelle in den ersten drei Tagen nach der Planung um 75 Prozent sinken kann, [165] kann mithilfe der SML-Modelle die Abweichung der Prognosen für den gesamten Planungshorizont konstant

[165] vgl. Schuh et al., 2015

niedrig gehalten werden. Das GBR-Modell erzielt etwa eine mittlere prozentuale Abweichung (MAPE) von 23 Prozent. Dies weist eine deutliche Verbesserung gegenüber traditioneller Vorhersagemodelle der PPS auf.

6.2 Ausblick

Ansätze basierend auf den Algorithmen von künstlicher Intelligenz kommen aufgrund ihres großen Potentials immer häufiger in der Industrie zum Einsatz. Die Potentiale sind aber weitgehend ungenutzt. Um dem ständigen Wettbewerb und den Kundenanforderungen gerecht zu werden, muss die Planungssicherheit in der PPS gewährleistet werden. Die Implementierung von künstlicher Intelligenz in der PPS wird in den kommenden Jahren möglicherweise den entscheidenden Vorteil bringen.

Um die Planungssicherheit in der PPS zu erhöhen, wurde in dieser Arbeit ein Ansatz basierend auf maschinellem Lernen gewählt. Dabei wurden ausschließlich Machine Learning-Algorithmen aus der überwachten Lernmethode verwendet. Weitere Forschungsfelder könnten sich mit der Frage nach der Leistung anderer Lernmethoden, wie etwa dem unüberwachten Lernen, beschäftigen. Zudem könnten Techniken des Deep Learnings im Hinblick auf die Produktionsplanung untersucht werden.

Um die Prognosegenauigkeit der Modelle zu verbessern, wurde der Datensatz mithilfe von Feature Engineering vorbereitet. Hier könnten weitere Ansätze zur Verbesserung der Modelle herangezogen werden. Die Modelle könnten etwa mithilfe von sogenanntem *Hyperparametertuning* weiter verbessert werden. Dabei werden die Parameter der einzelnen SML-Algorithmen iterativ angepasst.

Das Ergebnis dieser Arbeit bietet ein Spektrum an ausgewählten SML- Algorithmen, die in der PPS am häufigsten zur Prognose von Zielgrößen eingesetzt werden können. Die Messung der Prognoseleistung wurde anhand eines Anwendungsfalls aus der Stahlindustrie überprüft. Die Arbeit kann etwa als Grundlage für die Implementierung von SML-Algorithmen in anderen Industrien herangezogen werden.

Zuletzt wurde im Rahmen der Ausarbeitung auf die Definition der Planungsqualität und ihrer Implementierung in die PPS aufgezeigt. Auf Basis der Ansätze von Ryback et al. aus Kapitel 3.4 kann unter Anwendung der in dieser Arbeit ausgewählten SML-Algorithmen die Planungsqualität als Leistungsindikator im Unternehmen eingeführt werden.

7 Anhang

7.1 Feature Engineering

7.1.1 Entfernen von Duplikaten

```
import pandas as pd

process_data = pd.read_excel("Prozesszeiten.xlsx")

process_data = process_data.drop_duplicates(subset=["FA_Nummer"],
                                            keep= "last")
```

7.1.2 Eliminierung unnützer Features

```
process_data.drop(["FA_Nummer","Auftrag","Woche.Plan",
                   "Vollständiger.Name","RA","KundAuft","Produktionsstart,
                   "Produktionsende","Qualität_x","Qualität_y","DateTime",
                   "BuchDatum","Erfaßt.am","Uhrz.RM","DateCol","TimeCol"],
                   axis = 1)
```

7.1.3 Quantifizierung fehlender Daten

```
missing_values_percentage = process_data.isnull().mean()
x_percent = 0.05

for feature in process_data.columns:
      if missing_values_percentage[feature] > x_percent:
                   process_data = process_data.drop(feature, axis = 1)
```

7.1.4 Identifizierung kategorischer und numerischer Features

```
col_cat = ["Marke","QuaVorsch","S","Vrg","ARBG","ArbPlatz", "SCHM"]

process_data[col_cat] = process_data[col_cat].astype(object)

cont = process_data.drop(col_cat, axis=1)

for j in cont.columns:
         cont[j] = cont[j].astype(float)
         process_data[j] = cont[j]
```

7.1.5 Training- Test Split

```
X = process_data.drop("Prozesszeit", axis = 1) #Eingabevektor
Y = process_data["Prozesszeit"] #Zielvektor

from sklearn.model_selection import train_test_split



x_train, x_test, y_train, y_test = train_test_split(X, Y,
                                            test_size = 0.3)
```

7.1.6 Umgang mit Ausreißern

```
from feature_engine.outliers import OutlierTrimmer

process_data["Prozesszeit/Outlier"] = process_data["Prozesszeit"]

capper = OutlierTrimmer(capping_method='quantiles',
                            tail='right',
                           fold= 0.1,
                           variables=["Prozesszeit/Outlier"])
x_cap_train = capper.fit(x_train)

x_cap_train = capper.transform(x_train)
x_cap_test = capper.transform(x_test)

x_train = x_train.drop("Prozesszeit/Outlier", axis = 1)
x_test = x_test.drop("Prozesszeit/Outlier", axis = 1)
```

7.1.7 Kategoriale Kodierer

7.1.7.1 Kodierung von seltenen Kategorien

```
from feature_engine.encoding import RareLabelEncoder

rare = RareLabelEncoder(tol=0.001,
                        variables =["Marke","QuaVorsch", "SCHM"])

rare.fit(x_train)

x_train = rare.transform(x_train)
x_test = rare.transform(x_test)
```

7.1.7.2 Ordinale Kodierung

```
from feature_engine.encoding import CountFrequencyEncoder

encoder = CountFrequencyEncoder()
encoder.fit(x_train,y_train)

x_train = encoder.transform(x_train)
x_test = encoder.transform(x_test)


# Entfernung von Beobachtungen aus x_test und y_test,
# die nicht kodiert werden konnten:

rows_with_nan = [index for index, row in x_test.iterrows() if
row.isnull().any()]
x_test.dropna(axis = 0, inplace=True)
y_test.drop(rows_with_nan, axis = 0, inplace = True)
```

7.1.8 Feature Selection

7.1.8.1 Auswahl der Features nach ihrer Einzelleistung

```
from feature_engine.selection import SelectBySingleFeaturePerformance

SelectBySingleFeaturePerformance(estimator = LM,
                                 Scoring= "neg_root_mean_squared_error",
                                      cv = 10,
                                      threshold = None)

x_train_select = tr.fit_transform(x_train, y_train.values.ravel())
x_test_select = x_test[x_train_select.columns]
```

7.1.8.2 Rekursive Feature Eliminierung

```
from feature_engine.selection import RecursiveFeatureElimination

RecursiveFeatureElimination(estimator = LM,
                            Scoring ="neg_root_mean_squared_error",
                            cv = 10,
                             threshold = 0.5)

x_train_elim = tr.fit_transform(x_train, y_train.values.ravel())
x_test_elim = x_test[x_train_elim.columns]
```

7.2 Modeling

```
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn import linear_model
from sklearn import svm
from sklearn.neural_network import MLPRegressor
from sklearn import tree

DT = tree.DecisionTreeRegressor()
LM = linear_model.LinearRegression()
SVR = svm.SVR()
MLP = MLPRegressor()
RF = RandomForestRegressor()
GBR = GradientBoostingRegressor()
```

```
def TrainModels(x_train,y_train):
    DT.fit(x_train, y_train.values.ravel())
    LM.fit(x_train, y_train.values.ravel())
    SVR.fit(x_train, y_train.values.ravel())
    MLP.fit(x_train, y_train.values.ravel())
    RF.fit(x_train, y_train.values.ravel())
    GBR.fit(x_train, y_train.values.ravel())
```

```
def Predict(x_test):
    predict_RM = DT.predict(x_test)
    predict_LM = LM.predict(x_test)
    predict_SVR = SVR.predict(x_test)
    predict_MLP = MLP.predict(x_test)
    predict_RF = RF.predict(x_test)
    predict_GBR = GBR.predict(x_test)
```

7.3 Evaluierung

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error

def MAE(prediction):
    return mean_absolute_error(y_test, prediction)

def MAPE(prediction):
    return mean_absolute_percentage_error(y_test, prediction)*100

def MSE(prediction):
    return mean_squared_error(y_test, prediction)

def RMSE(prediction):
    return mean_squared_error(y_test, prediction, squared = False)

def NRMSE(prediction):
    NRMSE = mean_squared_error(y_test,
prediction,squared=False)/(y_test["Prozesszeit"].max() -
y_test["Prozesszeit"].min())*100
    return NRMSE
```

7.4 Evaluierungsergebnisse der SML-Algorithmen

Ohne Feature Auswahl Methode						
Fehlerfunktion	**RT**	**LM**	**SVR**	**ANN**	**RF**	**GBR**
MAE	19,15	14,33	27,093	19076	13,2	12,71
MAPE	34,03	25,49	49,936	36122	23,8	22,85
MSE	672,8	581,3	1314	528824214	311,6	282,5
RMSE	25,94	24,11	36,249	22996	17,65	16,81
NRMSE	14,82	13,78	20,714	13140	10,09	9,605

Tabelle 11: Evaluierungsergebnisse ohne Feature Auswahl Methode

Feature Auswahl nach Einzelleistung						
Fehlerfunktion	**RT**	**LM**	**SVR**	**ANN**	**RF**	**GBR**
MAE	18,53	15,28	22,59	15,73	13,86	13,12
MAPE	31,51	27,25	42,05	29,80	24,97	23,56
MSE	662,67	486,60	909,51	426,15	346,72	300,66
RMSE	25,74	22,06	30,16	20,64	18,62	17,34
NRMSE	14,71	12,61	17,23	11,80	10,64	9,91

Tabelle 12: Evaluierungsergebnisse für die Feature Auswahl nach ihrer Einzelleistung

Rekursive Feature Auswahl						
Fehlerfunktion	**RT**	**LM**	**SVR**	**ANN**	**RF**	**GBR**
MAE	16,44	15,71	18,03	14,37	14,16	13,95
MAPE	28,46	27,81	31,10	25,75	24,99	25,17
MSE	553,39	418,03	623,80	353,03	366,08	339,89
RMSE	23,52	20,45	24,98	18,79	19,13	18,44
NRMSE	13,44	11,68	14,27	10,74	10,93	10,53

Tabelle 13: Evaluierungsergebnisse für die rekursive Feature Auswahl

7.5 Visualisierte Planungsqualität der SML- Algorithmen

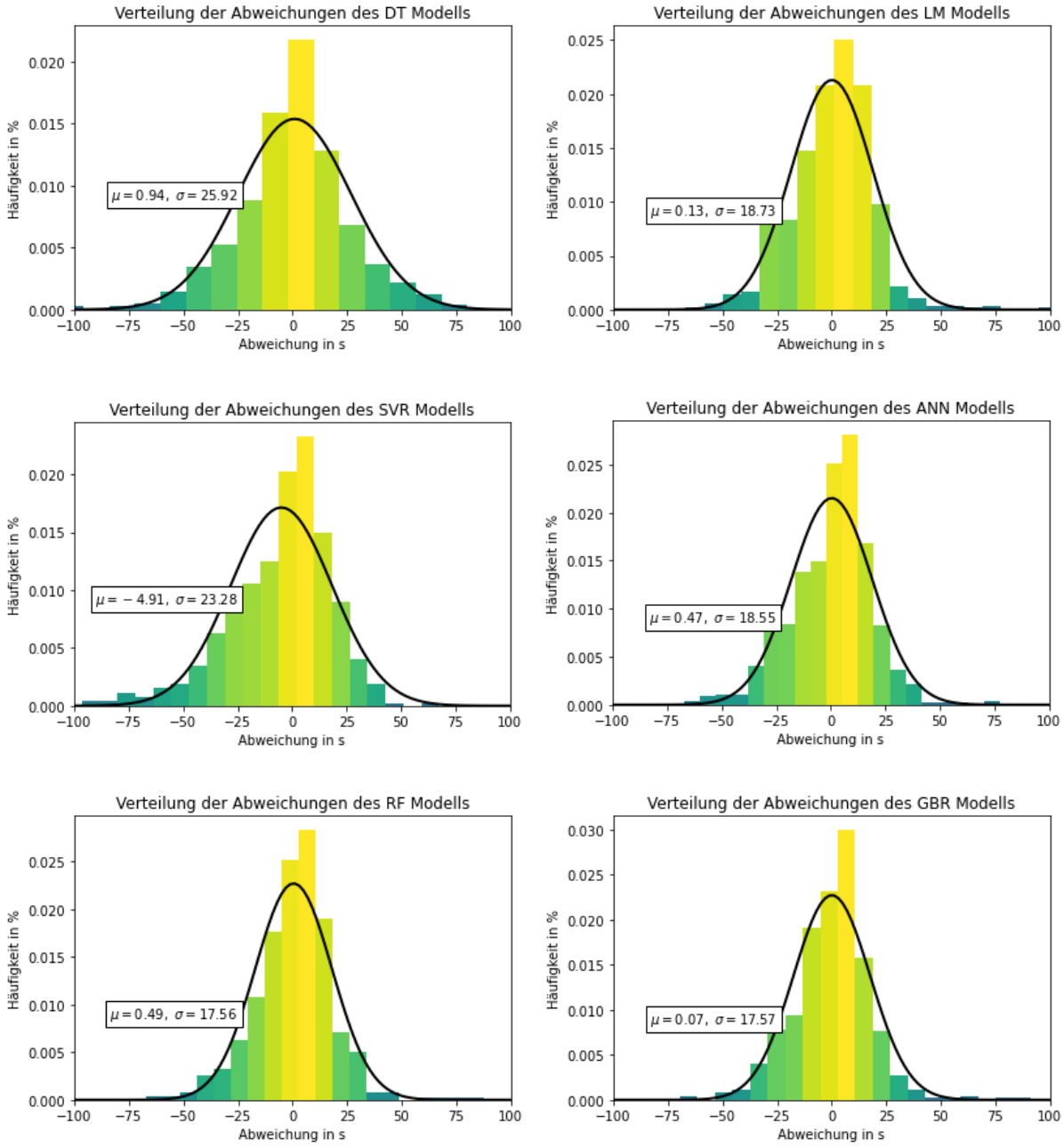

Tabelle 14:Visualisierte Planungsqualität der einzelnen SML-Algorithmen

8 Literaturverzeichnis

Awad, M., Khanna, R., Awad, M., Khanna, R., 2015. Support vector regression. Effic. Learn. Mach. Theor. Concepts Appl. Eng. Syst. Des. 67–80.

Bender, J., Ovtcharova, J., 2021. Prototyping Machine-Learning-Supported Lead Time Prediction Using AutoML. Procedia Comput. Sci. 180, 649–655.

Bishop, C.M., Nasrabadi, N.M., 2006. Pattern recognition and machine learning. Springer.

Cadavid, J. P. U., Lamouri, S., Grabot, B., & Fortin, A. (2019). Machine learning in production planning and control: A review of empirical literature. *IFAC-PapersOnLine*

Cellier, P., Driessens, K., 2020. Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I. Springer Nature.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., 2000. CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc 9, 1–73.

Dong, G., Liu, H., 2018. Feature engineering for machine learning and data analytics. CRC Press.

Duboue, P., 2020. The art of feature engineering: essentials for machine learning. Cambridge University Press.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

Gahm, C., Uzunoglu, A., Wahl, S., Ganschinietz, C., Tuma, A., 2022. Applying machine learning for the anticipation of complex nesting solutions in hierarchical production planning. Eur. J. Oper. Res. 296, 819–836.

Galli, S., 2022. Python feature engineering cookbook: over 70 recipes for creating, engineering, and transforming features to build machine learning models. Packt Publishing Ltd.

Galli, S., 2021. Feature-engine: A Python package for feature engineering for machine learning. J. Open Source Softw. 6, 3642.

Gardner, M.W., Dorling, S.R., 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos. Environ. 32, 2627–2636.

Gyulai, D., Pfeiffer, A., Monostori, L., 2017. Robust production planning and control for multi-stage systems with flexible final assembly lines. Int. J. Prod. Res. 55, 3657–3673.

Gyulai, D., Pfeiffer, A., Nick, G., Gallina, V., Sihn, W., Monostori, L., 2018. Lead time prediction in a flow-shop environment with analytical and machine learning approaches. IFAC-Pap. 51, 1029–1034.

Hackeling, G., 2017. Mastering Machine Learning with scikit-learn. Packt Publishing Ltd.

Jo, T., 2021. Machine Learning Foundations. Springer.

Johnston, B., Mathur, I., 2019. Applied supervised learning with Python: use scikit-learn to build predictive models from real-world datasets and prepare yourself for the future of machine learning. Packt Publishing Ltd.

Kempf, K., Uzsoy, R., Smith, S., Gary, K., 2000. Evaluation and comparison of production schedules. Comput. Ind. 42, 203–220.

Khair, U., Fahmi, H., Al Hakim, S., Rahim, R., 2017. Forecasting error calculation with mean absolute deviation and mean absolute percentage error, in: journal of physics: conference series. IOP Publishing, S. 012002.

Kirchner, J., & Meyer, S. (2022). Literaturrecherche. Wissenschaftliche Arbeitstechniken für die MINT-Fächer, 131-142.

Kriegel, H.-P., Borgwardt, K.M., Kröger, P., Pryakhin, A., Schubert, M., Zimek, A., 2007. Future trends in data mining. Data Min. Knowl. Discov. 15, 87–97.

Lingitz, L., Gallina, V., Ansari, F., Gyulai, D., Pfeiffer, A., Sihn, W., Monostori, L., 2018. Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer. Procedia Cirp 72, 1051–1056.

Louppe, G. (2014). Understanding random forests. Cornell University Library, 10.

Lödding, H., 2005. Verfahren der fertigungssteuerung. Springer.

Lucht, T., Mütze, A., Kämpfer, T., Nyhuis, P., 2021. Model-Based Approach for Assessing Planning Quality in Production Logistics. IEEE Access 9, 115077–115089.

Mahesh, B., 2020. Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet] 9, 381–386.

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. AI magazine, 27(4), 12-12.

Mitchell, T. M. (1997). Machine learning (Vol. 1, No. 9). New York: McGraw-hill.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). Foundations of machine learning. MIT press.

Ozdemir, S., Susarla, D., 2018. Feature Engineering Made Easy: Identify unique features from your dataset in order to build powerful machine learning systems. Packt Publishing Ltd.

Pfeiffer, A., Gyulai, D., Kádár, B., Monostori, L., 2016. Manufacturing lead time estimation with the combination of simulation and statistical learning methods. Procedia CIRP 41, 75–80.

Ray, S., 2019. A quick review of machine learning algorithms, in: 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon). IEEE, S. 35–39.

Richter, S. (2019). Statistisches und maschinelles Lernen. Berlin, Heidelberg: Springer Berlin Heidelberg. DOI

Ryback, T., Lukas, L., Gaal, A., Gallina, V., Gyulai, D., 2019. Improving the planning quality in production planning and control with machine learning.

Sammut, C., Webb, G.I., 2011. Encyclopedia of machine learning. Springer Science & Business Media.

Sarkar, D., Bali, R., Sharma, T., 2018. Practical machine learning with python. Probl.-Solvers Guide Build. Real-World Intell. Syst. Berkely Apress.

Schacht, S., & Lanquillon, C. (2019). Blockchain und maschinelles Lernen. Wie das maschinelle Lernen und die Distributed-Ledger-Technologie voneinander profitieren,.

Schmidt, M., Nyhuis, P., 2021. Wirkzusammenhänge zwischen den PPS-Hauptaufgaben und den logistischen Zielgrößen, in: Produktionsplanung und-steuerung im Hannoveraner Lieferkettenmodell. Springer, S. 67–89.

Schuh, G. (ed.). 2015: Ergebnisbericht des BMF-Verbundprojektes PROSense. Hochauflösende Produktionssteuerung auf Basis kybernetischer Unterstützungssysteme und intelligenter Sensorik.

Schröer, C., Kruse, F., Gómez, J.M., 2021. A systematic literature review on applying CRISP-DM process model. Procedia Comput. Sci. 181, 526–534.

Schuh, G., 2007. Produktionsplanung und-steuerung: Grundlagen, Gestaltung und Konzepte. Springer-Verlag.

Sharma, D., Kumar, N., 2017. A review on machine learning algorithms, tasks and applications. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 6, 2278–1323.

Shearer, C., 2000. The CRISP-DM model: the new blueprint for data mining. J. Data Warehous. 5, 13–22.

Singh, A., Thakur, N., Sharma, A., 2016. A review of supervised machine learning algorithms, in: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). Ieee, S. 1310–1315.

Tewari, K., Vandita, S., Jain, S., 2020. Predictive analysis of absenteeism in MNCs using machine learning algorithm, in: Proceedings of ICRIC 2019: Recent Innovations in Computing. Springer, S. 3–14.

Wang, H., Lei, Z., Zhang, X., Zhou, B., & Peng, J. (2016). Machine learning basics. Deep learning, 98-164.

Weber, F., 2020. Künstliche Intelligenz für Business Analytics. Springer.

Welsch, A., Eitle, V., & Buxmann, P. (2018). Maschinelles Lernen. HMD Praxis der Wirtschaftsinformatik, 55(2), 366-382.

Wirth, R., Hipp, J., 2000. CRISP-DM: Towards a standard process model for data mining, in: Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining. Manchester, S. 29–39.

Zheng, A., Casari, A., 2018. Feature engineering for machine learning: principles and techniques for data scientists. O'Reilly Media, Inc.

Zhou, Z.-H., 2021. Machine learning. Springer Nature.

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Abkürzungsverzeichnis

ANN	Neuronale Netzwerke
AutoML	Auto Machine Learning Modell
EN	Elastisches Netz
ERP	Enterprise Resource Planning
FA	Feature Auswahl
FAE	Auswahl der Features nach ihrer Einzelleistung
GBM	Gradient Boosting Modell
kNN	k-Nearest Neighbors
KRR	Kernel Ridge Regression
LM	Lineare Regression
LR	Lasso Regression
MAE	Mittlerer absoluter Fehler
MAPE	Mittlerer absoluter prozentualer Fehler
MARS	Multivariate adaptive Regressionssplines
MLP	mehrschichtiges Perzeptron
MLR	Mehrfache lineare Regression
MSE	Mittlerer quadratischer Fehler
PPS	Produktionsplanung und -Steuerung
PR	Polynomiale Regression
RF	Random Forest Modelle
RFA	Rekursive Feature Eliminierung
RMSE	Wurzel des mittleren quadratischen Fehlers
RR	Ridge Regression
RT	Regressionsbaummodelle
SGD	Stochastic Gradient Descent
SML	Supervised Machine Learning
SVM	Support Vector Machines
SVR	Support Vector Regression