Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Lukas Lühr

Eignung von maschinellen Lernverfahren im Kontext eines KDD Prozesses für biochemische Prozessdaten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Lukas Lühr

Eignung von maschinellen Lernverfahren im Kontext eines KDD Prozesses für biochemische Prozessdaten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Tim Tiedemann

Eingereicht am: 16. November 2017

Lukas Lühr

Thema der Arbeit
Eignung von maschinellen Lernverfahren im Kontext eines KDD Prozesses für biochemische Prozessdaten

Stichworte
Bio-Tech, Chargenbaum Analyse, Maschinelles Lernen, KDD, Complex Objects

Kurzzusammenfassung
In dieser Arbeit wird der Knowledge Discovery in Databases (KDD) Prozess exemplarisch anhand von Logfiles, welche im Rahmen eines biochemischen Fertigungsprozesses entstanden sind, durchlaufen. Diese Logfiles enthalten aus einfachen Daten aggregierte Strukturen, welche auch als Complex Objects bezeichnet werden. Es wird geprüft, inwieweit eine Bearbeitung der Complex Objects anhand des klassichen KDD Prozesses möglich ist. In dem Zuge werden auch verschiedene Ansätze des Maschinellen Lernens gegenübergestellt und bezüglich ihrer Leistungsfähigkeit auf den gegebenen Daten verglichen. Hierbei werden Entscheidungsbäume und verschiede Architekturen von neuronalen Netzen wie das Perzeptron, Multi Layer Perzeptron und Long short-term memory Netzwerke betrachtet.

Lukas Lühr

Title of the paper
Bio-Tech batch tree analysis using differing machine learning based approaches

Keywords
Bio-Tech, Batchtree analysis, Machine learning, KDD, Complex Objects

Abstract
In this work, the Knowledge Discovery in Databases (KDD) process is exemplarily executed on the basis of logfiles, which were created as part of a biochemical manufacturing process. These logfiles contain structures which are aggregated of simple data, these structures are called complex objects. It will be examined to what extent it is possible to apply the classic KDD process to these complex objects and what kind of result is to be expected. Therefor various approaches of machine learning are compared in terms of their performance on the given data. In particular, decision trees and some architectures of neural networks such as the perceptron, multi-layer perceptron and long short-term memory networks are considered.

Inhaltsverzeichnis

Listings

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Das aktuelle Aufleben der Themengebiete Big Data und künstliche Intelligenz wie in Abbildung 1.1 zu erkennen, wirft gerade durch die Kombination mit immer stärkerer und verfügbarer Berechnungshardware ein neues Licht auf die Problematik des Data Minings.

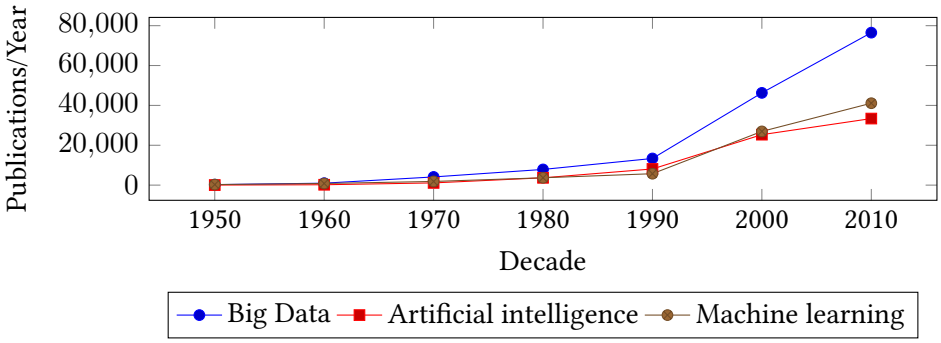

Abbildung 1.1: ACM Publikationen von 1950 bis 2010 [acm]

Es steht leistungsstarke Berechnungshardware zur Verfügung, welche sowohl die grob- als auch feingranulare Parallelisierung von Algorithmen ermöglicht. Zusätzlich gibt es erprobte Ansätze, welche in Form von Software Bibliotheken genutzt werden können, sodass eine Eigenimplementation nicht immer notwendig ist. Durch diese Gegebenheiten und die Tatsache, dass immer mehr Daten aus unterschiedlichsten Quellen zur Verfügung stehen, wird diese Thematik auch für kleinere Unternehmen oder Forschungsgruppen zugänglich.

So können beispielsweise Projekte, welche aufgrund ihrer Natur enorme Rechenleistung benötigen wie beispielsweise eine echtzeitfähige Berechnung von 3D Flüssigkeitssimulationen [TSSP16] heutzutage mit relativ geringen Hardwarekosten umgesetzt werden und bleiben somit nicht mehr nur Universitäten oder großen IT Riesen wie etwa Google, Amazon oder IBM vorbehalten.

Durch die Kombination von immer leistungsstärkerer Berechnungshardware und den stetig wachsenden Datenbeständen, welche heutzutage verfügbar sind, werden Ansätze des Maschi-

nellen Lernens (ML) und des Data Minings an sich immer attraktiver zur Verarbeitung dieser enormen Datenmengen.

Von herkömmlichen Berechnungsverfahren, den klassischen Algorithmen, unterscheiden sich Ansätze der künstlichen Intelligenz bzw. des Maschinellen Lernens dadurch, dass zur Lösung einer Aufgabestellung diese nicht direkt modelliert werden muss. Statt einer direkten Modellierung wird oftmals versucht, anhand von Beispieldaten ein Modell zu trainieren, um mehr oder weniger gut erkennbare Strukturen in den Daten zu entdecken und für die Lösung des Problems zu nutzen. Eben dieses Vorgehen ermöglicht es, mit ein und dem selben Verfahren eine Vielzahl von verschiedenen Problemen zu lösen, was gerade im Bereich des Knowledge Discovery in Databases (KDD) Prozesses sehr attraktiv ist. Bei dem KDD Prozess handelt es sich um ein Vorgehensmodell, welches ein mögliches Verfahren zum Bearbeiten und letztendlich Auswerten von Daten beschreibt. Der KDD Prozess in Kombination mit künstlicher Intelligenz wird in nahezu allen Branchen, welche einen Kontakt zur Informatik aufweisen, mehr oder weniger stark eingesetzt, eine Ausführung zu dem KDD Prozess findet sich in Abschnitt 2.1. Beispielsweise wird künstliche Intelligenz heutzutage bei der Vergabe von Krediten oder zur Analyse des Kaufverhaltens von (potentiellen) Kunden eingesetzt. In dem Unternehmen Werum IT Solutions GmbH, in welchem diese Bachelorarbeit entstanden ist, wird derzeit eine neue Abteilung für den Bereich Enterprise Manufacturing Intelligence aufgebaut, welche unter anderem in den Bereichen statistische Analysen als auch Data Mining Kompetenzen erweitern soll.

Aufgrund der steigenden Kundennachfrage und der stetig wachsenden Menge an Daten ist eine klassische Analyse auf Basis von statistischen Verfahren in absehbarer Zeit nur mit großem (personellem) Aufwand möglich. Aufgrund dieser Gegebenheit und auch gerade weil es sich bei den bereitgestellten Daten um nicht vor verarbeitete Realdaten aus dem Bereich der biochemischen Fertigungstechnik handelt, welche einen hohen Grad an Komplexität aufweisen, soll im Rahmen dieser Arbeit die Eignung des KDD Prozesses und einiger einfacher Data Mining Verfahren für diesen Aufgabenbereich evaluiert werden.

1.2 Ziele

Das Ziel dieser Arbeit ist es, mittels einer praktischen Abarbeitung die Eignung des KDD Prozesses für nicht vor verarbeitete Realdaten zu erproben, welche zusammengesetzte Daten wie etwa Bäume, enthalten. In dem Zuge wird ebenfalls geprüft, inwieweit mittels einfacher, maschineller Lernmethoden die Daten überhaupt ausgewertet werden können. Das Hauptaugenmerk liegt nicht auf der Entwicklung eines möglichst guten Modells, sondern auf der

Erprobung verschiedener Ansätze und dem Vergleich dieser, mit dem Ziel einen Grundstein für weiteres Arbeiten mit dem KDD Prozess und ML Verfahren zu setzen.

1.3 Weiterer Aufbau der Arbeit

Diese Arbeit teilt sich logisch in drei unterschiedliche Teile auf. Als Erstes folgt eine vorbereitende Analyse, in welcher die Aufgabenstellung klar definiert wird. Auch werden die theoretische Struktur der Daten sowie die einzelnen Phasen des KDD Prozesses vorgestellt.

Hierauf folgen praktische Experimente, in welchen die Phasen des KDD Prozesses sukzessiv durchlaufen werden. An dieser Stelle werden sowohl neuronale Netze als auch klassische Verfahren wie etwa C4.5 betrachtet sowie eine Gegenüberstellung durchgeführt, in welcher die Leistungsfähigkeit im Bezug auf die konkrete Problemstellung gegenübergestellt wird.

Anschließend wird diese Arbeit mit einer Zusammenfassung der Ergebnisse, einem Fazit sowie einem Ausblick zur möglichen Weiterführung oder Verwendung der Arbeit abgeschlossen.

2 Problemanalyse

Das Kapitel der vorbereitenden Analyse befasst sich in erster Linie mit den einzelnen Phasen des KDD Prozesses und dem KDD Prozess im Ganzen. Es werden Voraussetzungen und Ziele der einzelnen Phasen, sowie typisches Vorgehen beschrieben. Hierauf folgt eine kurze Vorstellung und Charakterisierung der ML Verfahren, welche im Rahmen dieser Arbeit betrachtet werden. Abgerundet wird das Kapitel mit einer Beschreibung der theoretischen Struktur der Datenbasis und einer Konkretisierung der Aufgabenstellung.

2.1 Der KDD Prozess / Analyse des KDD Prozesses

Der KDD Prozess beschreibt ein mögliches Vorgehen zur Gewinnung von Wissen aus Daten. Ziel des KDD Prozesses ist es, die wachsenden Datenmengen durch einen zumindest teilweise automatisierten Prozess zu bewältigen. Dieses Konzept wurde erstmals in [FPSS96] vorgestellt. Fayyad, Piatetsky-Shapiro und Smyth schrieben 1996: „KDD is the nontrivial process of identifying valid, novel, potantually useful, and ultimately understandable patterns in data “[FPSS96] . Es gibt direkte Alternativen zu dem KDD Prozess wie etwa das CRISP-DM Modell [CCK+00], welches durch ein wirtschaftliches Kollektiv erstellt wurde und heutzutage laut einer Umfrage der KDnugets Webseite eines der meist verwendeten Datamining Modell ist [CRI].

Im KDD Modell wird jedoch ein stärkeres Augenmerk auf die eigentliche Datenaufbereitung und -analyse gelegt. Eine detailliertere Ausführung des KDD Modells findet sich in [CL16] (dort in Kapitel 1.3, Ablauf der Datenanalyse).

Aufgrund eben dieses stärkeren Fokus auf der Aufbereitung und der Analyse wurde für diese Arbeit dieses Modell gewählt. Der KDD Prozess lässt sich in fünf gut trennbare Phasen aufteilen, welche jeweils die Transformation von einem (Zwischen-) Produkt wie etwa „Target Data “in ein Folgeprodukt, in diesem Falle „Preprocessed Data“, beschreiben. In der Literatur finden sich auch abweichende Definitionen oder Ansätze, welche den KDD Prozess auf weniger oder mehr Phasen abbilden, so wird er beispielsweise in [WH96] als vierstufiger Prozess dargestellt. In dieser Arbeit wird jedoch die fünfstufige Prozessdefinition verwendet, da sie ein gutes

Zwischenmaß an Granularität aufweist und dicht an der ursprünglichen Definition des KDD Prozesses liegt.

Die Phasen sind in Form eines Wasserfallmodells mit Rücksprung definiert (siehe Abbildung 2.1). Hierdurch ist es möglich, jede Phase des Modells beliebig oft zu durchlaufen, wodurch begangene Fehler beseitigt oder Ergebnisse verfeinert werden können. Es handelt sich also nicht um einen strikt linearen Prozess.

Eben dieser Prozess stammt jedoch aus einer Zeit, in welcher es sich bei Daten meist noch um primitive Typen wie Zahlen oder Texte gehandelt hat, die hier vorliegenden Daten bilden jedoch Complex Objects im Sinne von [ADM+92] ab. In der Veröffentlichung findet sich zu Complex Objects folgende Definition: „Complex objects are built from simpler ones by applying constructors to them. The simplest objects are objects such as integers, characters, byte strings of any length, booleans and floats (one might add other atomic types). There are various complex object constructors: tuples, sets, bags, lists, and arrays are examples. “[ADM+92].

Somit muss geprüft werden, inwieweit die Schritte des ursprünglichen KDD Prozesses modifiziert werden müssen, insbesondere in dem Punkt der Vorverarbeitung, um ein sauberes Arbeiten mit Complex Objects zu gewährleisten.

In den folgenden Abschnitten werden nun die einzelnen Prozessschritte des KDD Prozesses vorgestellt.

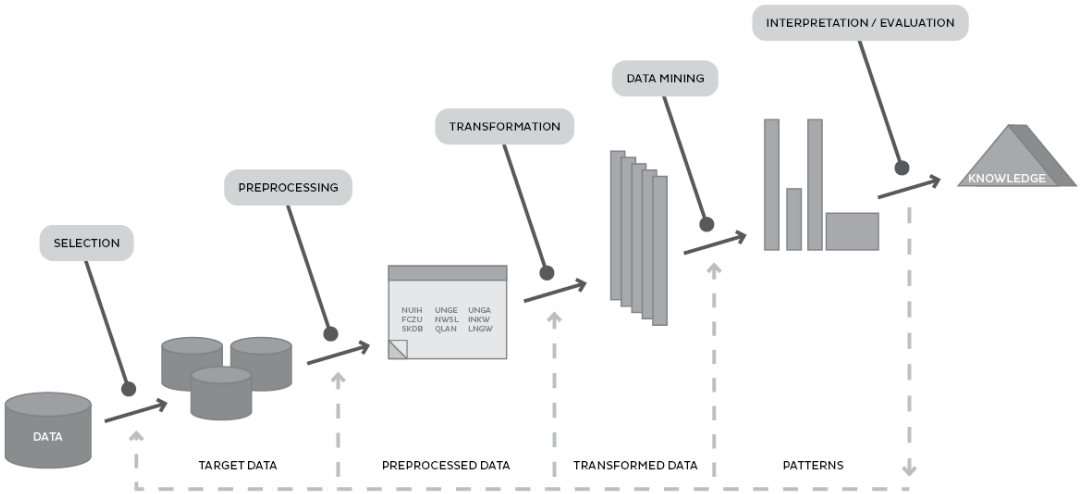

Abbildung 2.1: KDD Prozess, wie in [FPSS96] ausgeführt

2.1.1 Datenselektion

Die Datenselektion hat die Aufgabe, aus einem Datenbestand für eine Analyse relevante Daten zu extrahieren, die sogenannten Target Data. Hierfür ist es notwendig, das Ziel des anstehenden KDD Prozesses zu kennen. Denn abgesehen von einem rein explorativen Prozess, besteht meistens bereits zu Beginn die Anforderung, aus einer gewissen Menge von Daten, etwas konkretes zu ermitteln. Um dieses Ziel zu erreichen, müssen die vorliegenden Daten gesichtet und unter Anwendung von Domänenwissen bewertet werden[CL16].

Sollen beispielsweise die Motorleistungen von Kleinkrafträdern aus dem Datenbestand eines Fahrzeugherstellers analysiert werden, ist es sinnvoll, Informationen für Fahrzeuge einer anderen Klasse nicht im weiteren Analyseprozess zu betrachten, also gar nicht erst für die weitere Analyse auszuwählen.

Durch eine ungünstige Wahl in diesem Schritt kann es zu negativen Beeinträchtigung des weiteren Prozesses kommen, nicht jedoch nur durch das Vernachlässigen von relevanten Daten, sondern auch durch die Verwendung von nicht relevanten Daten. Wird beispielsweise das Ziel verfolgt, die Blüten von Blumen anhand ihrer geometrischen Struktur einer Spezies zuzuordnen, so ist es nicht sinnvoll oder gar kontraproduktiv etwa die Blütenfärbung in die Analyse einfließen zu lassen.

Durch eine Berücksichtigung der Farben kann es zwar möglicherweise zu einer besseren Unterscheidung kommen, jedoch entspricht dies nicht den Anforderungen, dass lediglich geometrische Eigenschaften betrachtet werden sollen.

Um diese Auswahl tätigen zu können, ist es in nicht trivialen Fällen notwendig, umfangreiches Wissen über die Anwendungsdomäne zu erlangen, um zwischen potentiell relevanten und unwichtigen Features, bzw. Teilen der Rohdaten unterscheiden zu können.

Ein weiterer Punkt, welcher sich gut an dieser Stelle des Prozesses eingliedern lässt, jedoch in dem ursprünglichem KDD Prozess nicht vorgesehen ist, ist die Datenanreicherung, wie in [BI06] beschrieben. Ziel dieses Schrittes ist es, die Daten um zusätzliche Informationen aus anderen Quellen anzureichern. Hierbei kommen je nach Aufgabendomäne unterschiedliche Informationsquellen in Frage, so können etwa Wetterdaten, Verkehrsinformationen, Zeitzoneninformationen oder personenbezogene Daten, etwa aus sozialen Netzwerken wie Facebook, in Frage kommen, um nur einige Beispiele zu nennen.

Expandiert man den Begriff der Datenbasis, welche in der Datenselektion des KDD Prozesses betrachtet wird, sodass diese auch fremde Datenquellen umfasst, so ist der Schritt der Datenanreicherung ein natürlicher Bestandteil der Datenselektion und somit vollständig in den KDD Prozess integriert.

Im Allgemeinen ist für eine Datenselektion auch zu beachten, dass Daten ggf. aus technischen oder rechtlichen Gründen nicht immer ohne weiteres in die Target Data überführt werden können. In diesem Fall ist der Vorgang unter Berücksichtigung der Limitationen zu wiederholen oder die rechtlichen Probleme zu klären.

2.1.2 Vorverarbeitung

Um auf den in der Datenselektion ausgewählten Daten spätere Analysen durchführen zu können, ist es oftmals notwendig, diese zu bereinigen bzw. eine gewisse Vorverarbeitung durchzuführen, um ihre Qualität zu steigern. Für eine Bereinigung von Daten ist es unter anderem notwendig, Strategien für den Umgang mit fehlerhaften oder fehlenden Daten aufzustellen, da diese Art von Daten den weiteren Verlauf des KDD Prozesses negativ beeinträchtigen können. Auch Probleme, wie etwa eine nicht einheitliche Struktur der Daten auf semantischer und syntaktischer Ebene, müssen an dieser Stelle behandelt werden. Neben dem Umgang mit solchen Gegebenheiten ist ein weiteres wichtiges Ziel der Vorverarbeitung die Steigerung der allgemeinen Datenqualität. Hierzu kann das Entrauschen von Messreihen über Filter gezählt werden. Doch abgesehen von der Behandlung von Problemen ist es auch nötig, Strategien zum Umgang mit speziellen Eigenschaften der Daten zu definieren, wie etwa der Umgang mit Zeitreihen in den Daten.

Einige häufige Probleme der Datenvorverarbeitung und entsprechende Lösungsansätze werden hier im Folgendem exemplarisch dargelegt und an Beispieldaten erklärt.

Fehlerhafte Daten

Um fehlerhafte Daten zu entdecken, ist oftmals ein hohes Maß an Domänenwissen erforderlich. Ein einfaches Beispiel, welches dies verdeutlicht, wäre beispielsweise ein personenbezogener Datensatz, in welchem Vor- und Zuname vertauscht wurden. Ein solcher Defekt lässt sich bei einem Namen wie „Christian Tim“ nicht mehr ohne weiteres erkennen. In anderen Fällen, wo beispielsweise in einer nicht normalisierten Datenbank für eine Materialnummer in zwei unterschiedlichen Datensätzen zwei unterschiedliche Bezeichnungen auftauchen, lässt sich der Defekt jedoch recht schnell mechanisch ermitteln. Auch unrealistische Messdaten wie etwa ein kurzzeitiger extremer Temperatur- oder Beschleunigungswert in einer Zeitreihe können in diese Kategorie fallen. Die Behandlung von derartigen Defekten in den Daten wird in dem Abschnitt 2.1.2 beschrieben.

Nach der Identifizierung eines solchen Defektes lassen sich einige verschiedene Methoden zur Beseitigung von eben diesem einsetzen wie beispielsweise in [CL16] beschrieben, jedoch

ist zu prüfen inwieweit eine Bereinigung im Falle von fehlerhaften Daten überhaupt sinnvoll oder notwendig ist, da auch in den Defekten Informationen enthalten sein können.

- Das gesamte Attribut zu ignorieren ist eine Möglichkeit, falls nicht ausreichend intakte Datensätze vorhanden sind. Dies kann jedoch auch zu Informationsverlust führen, ist daher nicht unbedingt als erstes Mittel anzuwenden.
- Eine manuelle Korrektur der Werte ist im Falle von wenigen betroffen Datensätzen oftmals eine gute Maßnahme, führt jedoch bei wachsender Anzahl von Fehlern zu einem enormen Arbeitsaufwand und ist somit auch nicht immer praktikabel.
- Handelt es sich bei dem Attribut, bei welchem Defekte auftreten, um ein metrisches Attribut, so ist es möglich, die fehlerhaften Werte durch eine Verrechnung der verbleibenden Werte zu ersetzen. Dieses Verfahren lässt sich noch weiter verfeinern. Handelt es sich etwa um eine Klassifikationsaufgabe, so kann für die Verrechnung jeweils nur die fragliche Klasse verwendet werden. Auch eine Verwendung des K-Neares Algorithmus, um Kandidaten für eine Verrechnung zu finden, ist ggf. sinnvoll. Bei etwa nominalen oder ordinalen Attributen ist ein ähnliches Vorgehen möglich, jedoch wird statt einer Berechnung meist das häufigste Element verwendet.
- Über Relationen in den Attributen des Datensatzes lassen sich unter Umständen die fehlerhaften Werte ermitteln. Ist etwa bei einem Eintrag, bei welchem Materialnummer und Name vermerkt seien sollen, lediglich die Nummer vermerkt, so kann bei einem anderen Eintrag, welcher beide Attribute enthält, oder über ein externes Mapping der fehlende Wert ermittelt werden. In diesem Falle sollte jedoch darüber nachgedacht werden, eines der beiden Attribute zu entfernen, da es sich offensichtlich um redundante Daten handelt.
- Die wohl einfachste Möglichkeit, das Problem fehlerhafter Daten zu behandeln, ähnlich dem Ignorieren eines Attributes, ist es, bei dem betroffenen Datensatz von der weiteren Verarbeitung abzusehen und diesen zu verwerfen. Sollten jedoch zu viele Datensätze davon betroffen sein, kann dies zu einem zu hohen Datenverlust führen.
- Da auch ein fehlerhafter Wert eine Bedeutung haben kann, wird ein solcher oftmals auch durch einen speziellen Markenwert ersetzt, welcher diesen Wert als fehlerhaft kennzeichnet, und dieser somit in der Analyse als solcher betrachtet werden kann. Dieses Vorgehen wäre etwa bei nicht leserlichen Kornrollunterschriften sinnvoll, da diese auch absichtlich so erstellt worden sein können.

Tabelle 2.1: Exemplarisch zu behandelnde Fehlerfälle

ID	Vorname	Nachname	Mitglied seit	Geburtsjahr
1	Sebastian	Müller	**1855**	1996
2	Heinz	Peter	1988	1960
3	Anja	**23**	2002	1980
4	Herbert	Meier	2001	1970

- Im Falle von Messreihen bieten sich auch Filter an, da es sich bei Werten mit starker Abweichung selten um Ausreißer handelt. Oftmals ergeben sie sich durch einen Messfehler. Wie jedoch in Abschnitt 2.1.2 beschrieben, kommt es durch die Verwendung von Filtern auch zu Informationsverlust.

Exemplarische Behandlung von fehlerhaften Daten In diesem Abschnitt werden anhand von fiktiven Daten zwei ausgewählte Verfahren zur Behandlung von fehlerhaften Daten durchgeführt, welche teilweise auch im praktischen Part der Arbeit Anwendung finden.

Zur Vorstellung der Verfahren werden diese auf die in Tabelle 2.1 abgebildeten Daten angewendet. Die Daten weisen zwei Defekte auf, in dem Datensatz mit der ID 1 ist ein Beitrittsjahr vor der Geburt angegeben, und im Datensatz mit der ID 3 wurde eine Zahl als Nachname eingetragen. Im Falle von Datensatz 1 ist es prinzipiell nicht eindeutig erkennbar, ob ein falsches Geburts- oder Beitrittsjahr angegeben wurde. Hierfür ist somit wieder Domänenwissen notwendig. Unter der Annahme, dass es sich bei der Datenbank um Kunden eines Online Shops handelt, kann mit Sicherheit gesagt werden, dass das „Mitglied seit“ Attribut des Datensatzes betroffen ist.

Entfernung defekter Attribute Durch die Entfernung von Attributen, welche einen Fehler aufweisen, ergibt sich die in Tabelle 2.2 gezeigte Situation. Durch diese Behandlung wurden auch sehr viele brauchbare Werte entfernt, was Zweifel an der Sinnhaftigkeit einer Entfernung von defekten Attributen an dieser Situation aufkommen lässt.

Entfernung defekter Datensätze Nach Entfernung von Datensätzen, welche einen Defekt aufweisen, ergibt sich die Datenbasis aus Tabelle 2.3. Wie auch bei der Entfernung von Attributen werden große Teile der Datenbasis verworfen. Hierdurch wird deutlich, dass ein solches Vorgehen nur bei einem geringen Anteil von Fehlern gut geeignet ist.

Tabelle 2.2: Exemplarisch zu behandelnde Fehlerfälle - Defekte Attribute entfernt

ID	Vorname	Geburtsjahr
1	Sebastian	1996
2	Heinz	1960
3	Anja	1980
4	Herbert	1970

Tabelle 2.3: Exemplarisch zu behandelnden Fehlerfälle - Defekte Datensätze entfernt

ID	Vorname	Nachname	Mitglied seit	Geburtsjahr
2	Heinz	Peter	1988	1960
4	Herbert	Meier	2001	1970

Fehlende Daten

Unter fehlenden Daten werden hier fehlende Ausprägungen von Attributen verstanden. Bei ihnen handelt es sich meist um eine spezielle Form von fehlerhaften Daten. Im Gegensatz zu den meisten fehlerhaften Daten lassen sich fehlende Daten mittels einfacher, oftmals automatisierter Methoden aufspüren. An erster Stelle ist jedoch zu ermitteln, ob es sich bei den fehlenden Daten überhaupt auch um fehlerhafte Daten handelt. Sollte bei einem Fertigungsprozess etwa eine Kontrollunterschrift nicht vorhanden sein, so kann dies zum Einen an korrupten Daten liegen, oder aber bedeuten, dass diese Kontrollunterschrift nicht erteilt worden ist, was wiederum eine wichtige Information sein kann. Der Umgang mit fehlenden Daten ist somit stark von der Problemdomäne abhängig, und für die Entscheidung, wie mit dieser Ausprägung von fehlerhaften Daten umgegangen werden soll, ist es oftmals sinnvoll einen Experten, welcher die Problemdomäne gut kennt, zu konsultieren. Handelt es sich jedoch wirklich um einen Fehler, lassen sich die allgemeinen Verfahren zur Beseitigung von fehlerhaften Daten weitgehend anwenden.

Entrauschen von Werten

Messreihen, welche die Entwicklung eines Wertes über einen zeitlichen Raum beschreiben, können je nach Datenquelle verrauscht sein. Das bedeutet, dass unrealistische Wertschwankungen vorliegen, welche wahrscheinlich die Aussagekraft der Daten verfälschen. Diese Schwankungen lassen sich beispielsweise über spezielle Verfahren zur Glättung der Daten beseitigen. Beispiels-

Tabelle 2.4: Beispielhafte Zeitreihe

ID					
1	Datum	2017-01-13	2017-01-19	2017-01-20	2017-01-25
	Wert	0.5	0.7	1.0	1.2
2	Datum	2016-11-10	2016-11-15	2016-11-17	2016-11-22
	Wert	0.4	0.8	1.2	1.5

Tabelle 2.5: Beispielhafte Zeitreihe - Bereinigt

ID					
1	Tag	0	6	7	12
	Wert	0.5	0.7	1.0	1.2
2	Tag	0	5	7	12
	Wert	0.4	0.8	1.2	1.5

weise ist die Verwendung eines Tiefpassfilters oftmals bereits ausreichend, um Messreihen zu entrauschen.

Jedoch gehen durch ein Entrauschen von Daten teilweise auch Informationen verloren. Prinzipiell muss bekannt sein, aus welcher Quelle die Daten stammen und ob bzw. wie stark diese überhaupt verrauscht sind, da dies nur auf Basis der Daten schwer bis gar nicht zu erkennen ist. Es muss also auch an dieser Stelle einiges an Domänenwissen vorhanden sein, um eine effektive Anwendung zu ermöglichen [CL16].

Zeitreihen Informationen

Handelt es sich bei Daten um Zeitreihen Informationen, ist es oftmals notwendig, diese weiter zu bearbeiten. Sind die zeitlichen Informationen etwa in Form von Zeitstempeln angegeben, kann es sinnvoll sein, diese in relative Werte in Bezug auf Abschluss bzw. Start der Zeitreihe umzuwandeln. Hierdurch werden Zeitreihen, welche zu unterschiedlichen Zeiten aufgezeichnet wurden, vergleichbar gemacht.

2.1.3 Daten Transformation

Die nun bereinigten Daten müssen, da die meisten ML Verfahren klar definierte Datenformate benötigen, in ein solches überführt werden. Verschiedene Verfahren haben unterschiedliche Anforderungen an die Daten, so können neuronale Netze beispielsweise grundsätzlich nur

auf numerischen Werten arbeiten, wohingegen beispielsweise der C4.5 Algorithmus auch mit nominalen bzw. ordinalen Werten umgehen kann. Intern wandelt der C4.5 Algorithmus sogar metrische Attribute in Intervalle, somit in ordinale Attribute um. Eine genauere Beschreibung des Verfahrens findet sich in [Qui93], das Verfahren wird auch in Abschnitt 2.2.2 beschrieben. An dieser Stelle ist also, je nach zu verwendender Data Mining Methode, eine entsprechende Umwandlung der Daten in ein anderes Format erforderlich.

Des Weiteren ist neben der strukturellen Anpassung von Daten auch die Reduktion des Feature Raumes, also eine semantische Anpassung, ein wichtiger Bestandteil der Daten Transformation.

Dieser Schritt wird meistens händisch durchgeführt, da er sich schlecht automatisieren lässt. Es gibt jedoch auch Ansätze diesen Teilschritt des KDD Prozesses stärker zu automatisieren, um den Menschen als Flaschenhals aus dem Prozess zu entfernen, wie in [HHK15] beschrieben. Eine vollständige Automatisierung wird im Rahmen dieser Arbeit jedoch nicht angestrebt, sodass dieser Teilschritt größtenteils manuell durchgeführt wird.

Im Folgenden werden häufig angewendete Verfahren der Daten Transformation dargelegt und beschrieben. Hierbei wird zwischen strukturellen, also syntaktischen, und semantischen Anpassungen unterschieden.

Syntaktische Anpassungen

Ziel einer syntaktischen Transformation ist es, die Struktur der Daten zu verändern, ohne dabei ihre Aussage zu verfälschen, um ein Verarbeiten mit Data Mining Methoden zu ermöglichen oder zu beschleunigen.

Typisches Vorgehen umfasst an dieser Stelle etwa eine Denormalisierung von Datenbanktabellen oder eine OneHot Kodierung von nominalen Daten. Auch das Entfernen von redundanten Attributen ist ein durchaus gängiges Vorgehen.

Diese Verfahren sind relativ unkompliziert und werden daher hier nicht weiter beleuchtet, sondern falls diese angewendet werden, kurz beschrieben. Inwieweit diese Art der Anpassung notwendig ist, wird im weiterem Verlauf der Arbeit geklärt.

Normalisierung und Standardisierung Zwei Verfahren, welche bei der Verwendung von neuronalen Netzen fast immer zwingend notwendig sind, weshalb sie hier auch nochmal kurz beleuchtet werden, sind die Standardisierung bzw. Normalisierung.

Bei der Verwendung von neuronalen Netzen ist es sinnvoll, alle Features auf einem einheitlichen Wertebereich abzubilden, um die Übergewichtung eines Features mit nummerisch höheren Werten zu verhindern [CL16].

Ein Verfahren, welches an dieser Stelle häufig verwendet wird, ist die Normalisierung, welche einen beliebigen Wertebereich auf den Bereich von 0...1 bzw. -1...1 abbildet. Grundsätzlich sieht die Formel für die Berechnung eines normalisierten Wertes n für den Wert x aus dem Raum y so aus : n = (x - min(y)) / (max(y)-min(y))

Diese Berechnung ist recht einfach durchzuführen, jedoch müssen immer dieselben min(y) und max(y) Werte verwendet werden, um bei gleicher Eingabe immer dieselbe Ausgabe zu erhalten. Das bedeutet, dass im Betrieb dieselben Werte verwendet werden müssen, welche auch zum Trainieren verwendet wurden.

Bei Extremwerten kommt es bei diesem Ansatz jedoch zu Problemen. Sind etwa 99% der Werte aus dem Raum y im Bereich von 0 .. 10 und 1% im Bereich von 1000..10000 mit einem Maximalwert von 10000, so werden 99% aller Daten auf 0.1% des Zielwertebereiches abgebildet.

Um dieses Problem zu umgehen, kann statt einer Normalisierung eine Standardisierung verwendet werden, welche auf einem Bereich von -inf...+inf, jedoch normal verteilt um 0 bzw. 0.5, mit einer Standartabweichung von 1 bzw. 0.5, abgebildet werden kann. Hierdurch wird in Kauf genommen, dass Werte außerhalb des Zielbereiches auftreten, jedoch wird der Einfluss von Extremwerten stark reduziert.

Semantische Anpassungen

Im Gegensatz zur syntaktischen wird durch eine semantische Anpassung die Aussage der Daten verändert. Häufiges Ziel dieses Prozesses ist die Dimensionsreduktion des Featurespaces, mit dem eigentlichen Ziel, die Mining Verfahren zu beschleunigen, oder ihre Ergebnisqualität zu steigern. Um dies zu erreichen, gibt es viele verschiedene Ansätze, welche verfolgt werden können. Einige Beispiele werden im folgenden Abschnitt kurz angerissen.

Binning Das Verfahren des Binnings kann dazu genutzt werden, beliebige metrische Attribute in ordinale Attribute umzuwandeln. Hierzu werden beliebig viele Intervalle auf dem gesamten Wertebereich des Attributes festgelegt, und diese mit einer Bezeichnung versehen. Die Ausprägungen des metrischen Attributes werden nun durch die Bezeichnung des Intervalles ersetzt, in welchem sie liegen [CL16].

Ziel ist es hierbei, Daten zu gruppieren, dies kann verwendet werden um etwa leicht verrauschte Werte zu bereinigen, um die Auswirkungen der Schwankungen zu verringern.

2.1.4 Data Mining

Der gesamte bisherige KDD Prozess arbeitet letzten Endes auf den Data Mining Schritt zu. Ziel dieses Schrittes ist es, aus den nun vor verarbeiteten und transformierten Daten ein

Modell abzuleiten, welches entsprechend den Anforderungen Aussagen bzw. Muster generieren kann. An dieser Stelle gibt es verschiedenste Probleme und Ansätze, um die Probleme zu lösen. Die Problemstellungen, welche am häufigsten auftreten, sind das Clustering, das Finden von Klassen in Daten; die numerische Vorhersage (Regression), welche die Vorhersage eines beliebigen numerischen Wertes auf Basis der Eingaben bezeichnet; und die Klassifikation. Bei der Klassifikation wird ähnlich wie bei dem Clustering auf Basis der Eingabe ein Datensatz einer Klasse zugeordnet, jedoch sind die Klassen im Gegensatz zum Clustering bereits vorher bekannt und müssen nicht erst durch das Verfahren ermittelt werden [CL16]. Einige ausgewählte, für diese Arbeit relevante Vertreter der Ansätze werden in Abschnitt 2.2 vorgestellt.

An dieser Stelle werden bereits Metriken für die Modelle benötigt, um eben diese Modelle zu optimieren. Jedoch sollten an dieser Stelle lediglich mechanische Verfahren ohne Nutzerinteraktion verwendet werden. Gründe hierfür sind unter anderem, dass eine manuelle Bewertung viel zu zeitintensiv ist, und dass sich durch eine Interaktion mit Menschen Fehler einschleichen können, welche in den Vorurteilen des Betrachters begründet sind.

Data Mining Metriken

Um ein Modell verbessern, also optimieren zu können, ist es notwendig eine Bewertung durchzuführen, um die Verbesserung ermitteln zu können.

Hierfür gibt es unterschiedlichste Maße, welche auf unterschiedliche Verfahren angewendet werden können. Ein sehr einfaches, welches für Klassifikatoren geeignet ist, ist etwa die Inkorrektheitsrate, welche als Verhältnis zwischen Klassifikationsfehlern und der Anzahl an durchgeführten Klassifikationen definiert ist.

Im Folgenden wird für die Bewertung der Verfahren das einfachen Konzept der Inkorrektheitsrate, bzw. Korrektheitsrate verwendet. Ergänzend wird auch der Recall-Wert der beiden Klassen in dem Interpretationsschritt des KDD Prozesses herangezogen. Der Recall-Wert beschreibt, wie viele Beispiele einer Klasse auch dieser zugeordnet wurden.

Oftmals werden mehrere Gütemaße ergänzend zueinander angewendet. Weitere Ausführungen finden sich hierzu in [CL16] in dem Kapitel neun, Bewertung.

Overfitting

Ein sehr häufiges Problem für Modelle, welche aus Daten generiert wurden, ist oftmals das sogenannte Overfitting, zu deutsch Überanpassung. Das Phänomen tritt auf, wenn ein Modell so stark an die Daten angepasst ist, auf deren Basis es entwickelt wurde, dass es zwar auf diesen sehr gut arbeiten kann, jedoch für bis dato ungesehene Daten schlechtere Ergebnisse liefert.

Es gibt mehrere potentielle Ursachen für diese Problematik, so kann etwa ein zu komplexes Modell bei einfachen oder wenigen Daten dazu neigen, die Trainingsdaten lediglich auswendig zu lernen, ohne die internen Strukturen zu erkennen. Dies führt zwar zu einem sehr geringen Fehler während des Trainings jedoch zu potentiell unerwünschten Verhalten auf neuen Daten [CL16].

Es gibt mehrere geeignete Möglichkeiten gegen Overfitting vorzugehen, so kann beispielsweise versucht werden, die Komplexität des Modells zu reduzieren, bei einem neuronalem Netz würde dies eine Reduktion der Neuronenzahl bedeuten. Andererseits kann die Anzahl der Trainingsdaten erhöht werden.

Im Gegensatz zum Overfitting ist auch ein Underfitting, eine Unteranpasung, möglich. Dies tritt etwa auf, falls das Modell für die Daten nicht ausreichend komplex ist. Das Modell ist nicht in der Lage die Struktur der Daten zu erlernen [CL16].

2.1.5 Interpretation

Der finale Schritt des KDD Prozesses ist die Interpretation der Ergebnisse des Data Mining Schrittes. Hierbei gilt es, den Erfolg des Prozesses zu bewerten, also den Wert der Ergebnisse des Data Mining Schrittes zu ermitteln.

Da der KDD Prozess nicht dem Selbstzweck, sondern der Unterstützung von Prozessentscheidungen dienlich sein soll, ist dieser Schritt unentbehrlich. Die Ergebnisse sollen vernünftige Entscheidungen im betrieblichem Kontext, unterstützen oder bestenfalls aufzeigen. Daher müssen die Artefakte des Data Mining Schrittes bewertet werden, um zu ermitteln, inwieweit sie als Entscheidungsgrundlage geeignet sind.

Um den Wert eines Modells bzw. Musters zu erfassen, wurden im Laufe der Zeit verschiedene Metriken definiert. Grundsätzlich lassen sie sich in subjektive und objektive Metriken bzw. Mischformen dieser unterteilen.

Eine objektive Metrik wie beispielsweise die Fehlerrate ist für eine allgemeine kundenunspezifische Bewertung von Modellen geeignet. Sie ist auf Klassifikationsproblemen als Verhältnis zwischen fehlerhaften Klasifikationen und gesamt durchgeführten Klassifikationen definiert. Jedoch wird hierdurch nicht der eigentliche Wert für den Kunden erfasst. Ein Modell mit einer verschwindend geringen Fehlerrate kann trotz der hohen Genauigkeit für einen Kunden vollkommen wertlos sein, wenn die Vorhersage für den Kunden uninteressant ist [PSM94].

Um eben diesen kundenspezifischen Wert, welcher im Rahmen des KDD Prozesses wesentlich relevanter ist, zu erfassen, können objektive Metriken verwendet werden. Sie werden hierbei aus der Sicht des Kunden, oder besser noch durch den Kunden angewendet.

Zwei Metriken, welche sich in diesem Rahmen für die Bewertung von Ergebnissen eignen, wurden 1994 in [PSM94] eingeführt. Die eine ist die Unexpectedness Metrik, die besagt, dass ein Muster interessant ist, wenn es den Benutzer überrascht. Die zweite Metrik, welche sich mit der ersten kombinieren lässt, ist die sogenannte Actionbility Metrik. Sie besagt, dass ein Muster dann für einen Benutzer interessant ist, wenn es ihm ermöglicht, für ihn vorteilhaft zu handeln.

Sowohl die Unexpectedness als auch die Actionbility Metrik sind beide recht objektive Metriken, insbesondere dies ist eine ihrer Stärken, da der Wert eines Musters stark vom Standpunkt des Betrachters abhängig ist. [ST95]

Neben diesen zwei Metriken gibt es noch viele weitere. Eine, welche für diese Arbeit ebenfalls herangezogen wird, ist die Verständlichkeits Metrik wie etwa in [CL16] beschrieben. Hierüber ist zu bewerten, inwieweit eine Aussage für den Anwender verständlich ist. Dies ist zwar bereits implizit in der Actionbility Metrik enthalten, wird an dieser Stelle aufgrund der hohen Bedeutung aber explizit erwähnt.

Beierle schrieb in [BI06], dass der KDD Prozess nur dann erfolgreich ist, wenn die von ihm gelieferten Ergebnisse, drei Kriterien erfüllen. Es muss sich bei ihnen um neues, nützliches und interessantes Wissen in verständlicher Form handeln. Diese Anforderungen sind stark subjektiv.

Die Bewertung kann somit nicht ohne ausgeprägtes Domänenwissen oder Kontakt mit dem Kunden bzw. Benutzer geschehen, von daher sind an dieser Stelle ggf. Interviews mit einem Experten für den Fertigungsprozess notwendig. Auch profitiert der Prozess an dieser Stelle von grafischen Darstellungen der Ergebnisse, da diese von einem Menschen wesentlich schneller als tabellarische Repräsentationen überblickt werden können.

2.2 Data Mining Methoden

Der Data Mining Schritt des KDD Prozesses erfordert die Anwendung eines oder mehrerer geeigneter Data Mining Verfahren. Hier finden beispielsweise Algorithmen wie C4.5 für Entscheidungsbäume oder künstliche neuronale Netze Anwendung. In diesem Abschnitt werden die für diese Arbeit relevanten Verfahren kurz vorgestellt und eingeordnet.

Grundsätzlich sind zwei verschiedene Data Mining Ansätze für diese Arbeit interessant, einerseits Klassifikation und andererseits Regression. Sie werden im Folgenden kurz vorgestellt. Alle Ansätze, welche hier verwendet werden, werden als Superviced Lerning Ansätze betrieben. Dies bedeutet, dass sie anhand von Beispieldaten lernen, für welche bereits Vorhersagewerte vorliegen [CL16].

Regression

Ziel eines Regressionsverfahrens ist es, einen Zusammenhang zwischen den bekannten Features und einem Zielwert zu modellieren. Hierzu wird eine Regressionsfunktion verwendet. Das konkrete Ziel dieses Ansatzes ist es, den Fehler zwischen Vorhersage und tatsächlichem Wert zu minimieren, um eine möglichst gute Vorhersage zu ermöglichen. Dieser Ansatz wäre somit zur Ermittlung des numerischen Wertes, auf Basis dessen die Klassen der einzelnen Datensätze ermittelt wurden, geeignet.

Ziel dieser Arbeit ist es jedoch, verschiedene Klassifikationsverfahren zu erproben, sodass dieser Ansatz im Folgenden nicht untersucht wird.

Klassifikation

Im Gegensatz zu Regressionsansätzen verfolgt die Klassifikation nicht den Ansatz einen numerischen Wert vorherzusagen, sondern den jeweiligen Daten Klassen zuzuordnen. Die Anzahl an Klassen ist hierbei endlich, und kann somit auch als Klassifikation auf unendlich vielen Klassen betrachtet werden.

Im Konkreten werden in dieser Arbeit verschiedene Klassifikationsverfahren verwendet, um die Chargenbaum Daten in die zwei Klassen „überdurschschnittliche Qualität“ und „unterdurchschnittliche Qualität“ einzuteilen.

Hierzu werden mehrere, verschiedene Verfahren aus dem Bereich des Maschinellen Lernens verwendet.

2.2.1 Erklärbarkeit von Mining Verfahren

Laut gängiger Meinung ist es zusätzlich auch möglich, Mining Verfahren in erklärbare und nicht erklärbare Verfahren zu unterteilen.

Zu den erklärbaren Verfahren werden Ansätze gezählt, welche ihre Entscheidung begründen. Einer ihrer Vertreter ist der C4.5 Algorithmus, welcher Entscheidungsbäume generiert, die als Fragenkatalog für die Entscheidung angesehen werden können. Andere Verfahren wie etwa neuronale Netze lassen sich bei den nicht erklärbaren Verfahren einordnen. Hiermit ist die Nachvollziehbarkeit eines Modells und somit die Ermittlung von für Entscheidungen zu Grunde liegenden Faktoren gemeint.

Vertreter von erklärbaren Data Mining Verfahren sind somit Modelle wie Entscheidungsbäume oder andere Regel generierende bzw. basierte Ansätze. Im Kontrast hierzu stehen Verfahren wie neuronale Netze oder Support Vektor Maschinen, welche zu den nicht erklärbaren Verfahren gehören.

Der grundlegende Unterschied zwischen diesen zwei Klassen ist die Tatsache, dass aus einigen Modellen keine oder nur unter sehr großen Aufwänden Rückschlüsse auf die für eine Entscheidung grundlegenden Faktoren möglich sind, bei neuronalen Netzen beispielsweise über eine Visualisierung der internen Variablen.

Obwohl im Falle der erklärbaren Verfahren die der Entscheidung zugrunde liegenden Faktoren bekannt sind, bedeutet dies jedoch nicht, dass es immer problemlos möglich ist, die Entscheidung konkret zu begründen. Ist das Modell entsprechend komplex, ist es auch hier nur schwer möglich, sinnvolles Wissen, abgesehen von der Vorhersage, aus dem Modell zu extrahieren.

Augenscheinlich sind somit nicht erklärbare Ansätze von Nachteil, da sie keine Rückschlüsse auf die Faktoren zulassen, welche etwa bei einem Fertigungsprozess zu minderwertiger Qualität führen und somit eine Optimierung nur bedingt möglich ist. Jedoch sind nicht erklärbare Verfahren oftmals leistungsstärker, was ihre Vorhersagegenauigkeit und ihr Abstraktionsvermögen angeht.

Aktuelle Forschung im Bereich von neuronalen Netzen arbeitet daran diese Modelle in den Bereich der Erklärbarkeit zu verrücken, ein Beispiel hierfür findet sich etwa in [dar].

2.2.2 Konkrete Data Mining Verfahren

In diesem Abschnitt werden die in dieser Arbeit betrachteten Mining Verfahren vorgestellt, es werden Vertreter von erklärbaren und nicht erklärbaren ML Verfahren gewählt.

C4.5 Algorithmus

Bei dem C4.5 Algorithmus handelt es sich um ein Verfahren, welches auf Basis von Beispieldaten einen Entscheidungsbaum generieren kann. Dieses konkrete Verfahren wird neben anderen Ansätzen im Rahmen dieser Arbeit verwendet. Um auf Basis des Entscheidungsbaumes Vorhersagen treffen zu können, ist es entweder notwendig, dass die Beispieldaten den gesamten Wertebereich aller möglichen Eingangsdaten abdecken. In diesem Falle könnte man jedoch einfach in der Liste aller Beispiele nach dem zu bewertenden Datensatz suchen. Oder es ist notwendig, dass auf Basis der Beispieldaten auf die zu bewertenden Datensätze geschlossen werden kann.

Die in Tabelle 2.6 gegebenen Daten definieren auf Basis der Wettervorhersage und des aktuellen Wetters, ob Kinder draußen spielen sollten. Diese Daten können nun in Kombination mit dem C4.5 Algorithmus verwendet werden, um einen beispielhaften Entscheidungsbaum zu generieren, um bei nicht in der Tabelle enthaltenden Situationen eine geeignete Vorhersage

Tabelle 2.6: Entscheidungstabelle - Wetter Beispiel [CL16]

Vorhersage	Temperatur	Luftfeuchtigkeit	Windig	Draußen spielen
Sonnig	Warm	Hoch	Nein	Nein
Sonnig	Warm	Hoch	Ja	Nein
Sonnig	Mild	Hoch	Nein	Nein
Sonnig	Mild	Normal	Ja	Ja
Sonnig	Kalt	Normal	Nein	Ja
Bewölkt	Warm	Hoch	Nein	Ja
Bewölkt	Warm	Normal	Nein	Ja
Bewölkt	Mild	Hoch	Ja	Ja
Bewölkt	Kalt	Normal	Ja	Ja
Regnerisch	Mild	Hoch	Nein	Ja
Regnerisch	Mild	Normal	Nein	Ja
Regnerisch	Mild	Hoch	Ja	Nein
Regnerisch	Kalt	Normal	Nein	Ja
Regnerisch	Kalt	Normal	Ja	Nein

treffen zu können, oder die den Daten zugrunde liegenden Regeln zu extrahieren. Bei Listing 2.2 handelt es sich um einen aus den Daten generierten Entscheidungsbaum. Das Ziel des C4.5 Algorithmus ist es jedoch nicht nur, einen Entscheidungsbaum zu generieren, sondern einen möglichst kleinen Entscheidungsbaum mit möglichst starken Fragen zu erstellen. Eine Frage ist stark, wenn die Verteilung der Klassen in den Ergebnismengen der Frage weniger gleichmäßig ist als in der Ursprungsmenge. Über diese Definition lässt sich die Qualität einer Frage ermitteln, und somit eine optimale finden. Eine Ausführung des C4.5 Algorithmus und des Konzeptes von Entscheidungsbäumen findet sich in [Qui93].

Die Abbildung 2.2 zeigt einen Entscheidungsbaum, welcher auf Basis der Daten aus Tabelle 2.6 generiert wurde. Es ist zu erkennen, dass das Zielattribut „Draußen spielen“ bereits mit sehr wenigen Fragen erfolgreich vorhergesagt werden kann.

Künstliche neuronale Netze

Bei künstlichen neuronalen Netze handelt es sich um ein Berechnungskonstrukt, welches einen universellen Funktion Aproximator abbilden kann[Csá01]. Es kann also eine beliebig komplexe mathematische Funktion in Form eines neuronalen Netzes abgebildet werden.

Die ersten Versuche in dieser Richtung wurden 1943 in [MP43] vorgestellt. Seitdem wurden die Konzepte und Modelle weiter entwickelt und durchliefen mehrere Blütephasen, gerieten jedoch immer wieder in den Hintergrund. In den letzten Jahren erlebten sie aufgrund von

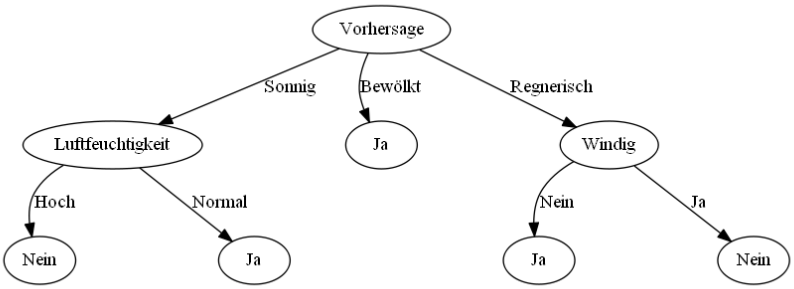

Abbildung 2.2: Aus Tabelle 2.6 generierter Entscheidungsbaum

immer stärkerer Berechnungshardware eine Renaissance. Eine übersichtliche Ausführung der frühen Geschichte von neuronalen Netzen findet sich in [sta]

Grundsätzlich basieren künstliche neuronale Netze auf künstlichen Neuronen, dessen Ausgabe in andere Neuronen einfließen kann. Die Berechnung der Aktivierung eines Neurons, also seiner Ausgabe, erfolgt in zwei Schritten. Als erstes fließen alle eingehenden Aktivierungen, sei es von anderen Neuronen oder von außerhalb des Netzes, oftmals in Kombination mit einer Gewichtung und einem BiosWertes, welcher die Aktivierung des Neurons in eine Richtung beeinflusst, ein. Das Ergebnis dieser Berechnungsfunktion wird als Nettoeingabe bezeichnet. Anschließend wird eine Aktivierungsfunktion auf diese Nettoeingabe angewendet. Bei dieser Funktion handelt es sich um eine stetig differenzierbare Funktion. Häufig werden an dieser Stelle die Sigmoid Funktion oder ähnliche verwendet. Diese Aktivierungsfunktion errechnet nun aus der Nettoaktivierung die Aktivierung des Neurons, die sättige Differenzierbarkeit ist an dieser Stelle lediglich für das Rückpropagieren von Fehlern, wie es beim Lernen erfolgt, notwendig. Ein Überblick und Vergleich über verschiedene Aktivierungsfunktionen wird in [DS93] gegeben.

Die konkrete Anordnung der einzelnen Neuronen wird als Architektur bezeichnet, hier gibt es wiederum viele verschiedene Möglichkeiten, die Neuronen anzuordnen. In dieser Arbeit werden jedoch lediglich drei Architekturen betrachtet, namentlich Perzeptron, Multi Layer Perzeptron und Long short-term memory Netzwerk (LSTM). Diese Architekturen werden im Folgenden kurz vorgestellt.

Perzeptron Bei einem Perzeptron handelt es sich um ein neuronales Netz, welches die einfachste mögliche Architektur aufweist. Zwischen der Eingabe und der eigentlichen Ausgabeschicht befindet sich keine weitere Verarbeitungsschicht. Diese Architektur ist die erste

Architektur für künstliche neuronale Netze und wurde in [MP43] vorgestellt. Jedoch wurde kurz nach Vorstellung dieser Architektur aufgezeigt, dass sie nicht in der Lage ist, ein nicht lineares Problem wie beispielsweise die Xor Funktion zu erlernen. Dieses Problem wurde 1969 in [MP69] dargelegt, und basiert auf der nicht linearen Separierbarkeit des Feature Spaces. Diese Erkenntnis führte zu einem zeitweiligem Stopp der Forschung an künstlichen neuronalen Netzen.

Multi Layer Perzeptron Bei einem Multi Layer Perzeptron (MLP) handelt es sich um eine Erweiterung des einfachen Perzeptrons, es verfügt über mindestens eine sogenannte Hidden Layer, Schichten von Neuronen zwischen Eingabe und Ausgabeschicht. Die Breite der Hidden Layer ist dabei irrelevant. Prinzipiell kann es als eine Anreihung von mehreren Perzeptronnetzwerken betrachtet werden, wodurch es in der Lage ist, nicht lineare Problemstellungen abzubilden.

LSTM Bei einem LSTM handelt es sich um eine eher junge und im Vergleich zu einem Perzeptron oder MLP doch eher exotischere Architektur, welche 1997 erstmals in [HS97] vorgestellt wurde, und durch eine Feedbackschleife in der Lage ist, Informationen zu einem späterem Zeitpunkt zu verwenden. Dies ist nicht der erste Ansatz künstliche neuronale Netzwerke mit einer Rückkopplung zu modellieren. Bereits 1982 wurde von J.J. Hopfield in [Hop82] ein ähnliches Konzept präsentiert. Durch spezielle Erweiterungen in der Architektur ist diese im Gegensatz zu früheren Ansätzen in der Lage, Informationen über einen langen Zeitraum zu speichern und ist somit gerade für weak-signal Erkennung und lange Zeitreihen gut geeignet.

Bei den hier zu analysierenden Daten handelt es sich zwar nicht direkt um eine Zeitreihe, jedoch lässt die Baumstruktur sich auf eine eben solche abbilden. Knoten, die dichter an der Wurzel, also dem Endprodukt sind, tauchen in der Zeitreihe später auf, da sie zu einem späterem Zeitpunkt in die Produktion einfließen. Dieses Konzept wird in Abschnitt 3.5 erneut aufgegriffen und umgesetzt.

2.3 Struktur der Datenbasis laut Dokumentation

Bei den Daten, welche in dieser Arbeit bearbeitet werden sollen, handelt es sich um Logfiles aus dem Produktivbetrieb in der pharmazeutischen Fertigungsindustrie.

Diese Logfiles enthalten Complex Objects, welche es ermöglichen einen gesamten Fertigungslauf zu einem späteren Zeitpunkt nachzuvollziehen. Konkret handelt es sich bei den Complex Objects um ein Konstrukt, welches in der Industrie als Chargenbaum bekannt ist.

In diesem Abschnitt wird die interne Struktur der Datenbasis stark vereinfacht dargestellt um einen ersten Überblick zu vermitteln. Eine feingranulare Analyse der Struktur Datenbasis findet sich 3.3.1.

Auch wenn die Bezeichnung für diese Struktur suggeriert, es handele sich um einen Baum, ist dies nicht der Fall. Der Begriff bezeichnet einen schleifenlosen gerichteten Graphen (DAG), welcher den Materialfluss während des Fertigungsprozesses beschreibt. Jeder Knoten in dem Graphen beschreibt den Übergang von einem Material oder mehreren Materialien zu einem Zwischen- bzw. Endprodukt, also einen Fertigungsschritt. Hierbei werden jeweils wichtige Daten wie die Menge oder die konkrete Charge der einfließenden Materialien erfasst. Auch werden Zwischenfälle oder anderweitig relevante Ereignisse für jeden Produktionsschritt in dem Chargenbaum abgelegt.

Wichtig ist zu beachten, dass nicht unbedingt Zeitangaben in einem Chargenbaum enthalten sein müssen. Wie in dem Beispiel unter 2.3.1 auch zu erkennen ist, ist diese Information nicht zwingend notwendig, auch wenn Zeitstempel eine Sequenzialisierung des Baumes erleichtern würden.

2.3.1 Struktur eines Chargenbaumes

Ein Chargenbaum beschreibt den Fertigungsprozess bis zu einem ausgewähltem Zwischen- oder Zielprodukt, hierzu werden die Materialflüsse dargestellt. Um den Chargenbaum für ein beliebiges Produkt aufzubauen, werden alle direkt einfließenden Materialien sowie die Mengen mit welchen sie in die Fertigung einfließen ermittelt, und in einem gerichtetem Graphen eingetragen. Für alle einfließenden Produkte wird wiederum ein Chargenbaum aufgebaut und an der entsprechenden Stelle im Graphen eingetragen. Dies erfolgt beliebig tief, bis alle Knoten mit einem Eingangsgrad von Null in der Datenbasis keine einfließenden Materialien mehr haben, und der Chargenbaum somit vollständig aufgebaut ist.

Beispiel Chargenbaum

In diesem Abschnitt wird ein beispielhafter Chargenbaum schrittweise aufgebaut und grafisch dargestellt.

Zu betrachten ist ein Produkt mit der Bezeichnung „Zielprodukt“, für dessen Fertigung ein Chargenbaum dargestellt werden soll. Die zugehörige Datenbasis ist in der Tabelle 2.7 zu finden. In das Produkt sind drei andere Produkte jeweils zu unterschiedlichen Anteilen eingeflossen. Von dem Produkt „Zwischenprodukt_A“ wurden 524ml verwendet, von dem Produkt „Zwischenprodukt_B“ 2kg und von dem Produkt „Eingekauft_C“ 20mg. Anhand

Tabelle 2.7: Beispielhafte Chargenbaum Struktur

Quelle	Ziel	Menge
Eingekauft_D	Zwischenprodukt_D	190ml
Eingekauft_E	Zwischenprodukt_D	50g
Eingekauft_A	Zwischenprodukt_A	3kg
Eingekauft_B	Zwischenprodukt_B	324ml
Zwischenprodukt_D	Zwischenprodukt_B	200ml
Zwischenprodukt_A	Zielprodukt	524ml
Zwischenprodukt_B	Zielprodukt	2kg
Eingekauft_C	Zielprodukt	20mg

dieser Information lässt sich in der erste Stufe die in Abbildung 2.3 dargestellte grafische Repräsentation des Chargenbaumes aufbauen.

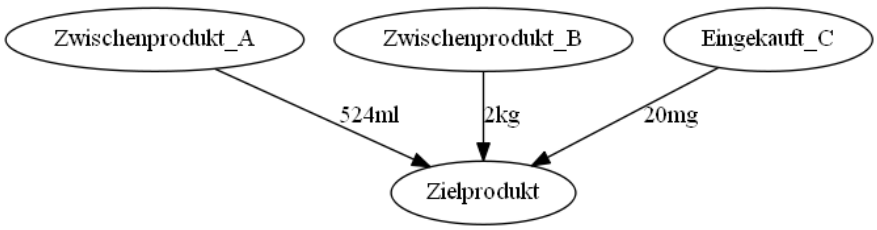

Abbildung 2.3: Chargenbaum auf Ebene eins

Im nächsten Schritt wird versucht, für die drei einfließenden Produkte einen eigenen Chargenbaum aufzubauen. Dabei ergibt sich das Zwischenprodukt_A, das aus 3kg des Produktes Eingekauft_A hergestellt wurde. Zwischenprodukt_B wurde aus zwei anderen Produkten gefertigt, aus 200ml Zwischenprodukt_D und aus 324ml Eingekauft_B. Für das Produkt Eingekauft_C gibt es keine weiteren Vorgänge, es handelt sich also bereits um ein Blatt im Chargenbaum. Eine grafische Darstellung des Baumes ist in Abbildung 2.4 gegeben.

Es ist nun für Eingekauft_A und für Zwischenprodukt_D wieder ihr Chargenbaum aufzubauen. Hierbei ergibt sich, dass Eingekauft_A keine Vorgänger hat, und es sich somit um ein Blatt handelt. Zwischenprodukt_D hat Vorgänger, Eingekauft_D fließt mit 190ml ein und Eingekauft_E fließt mit 50g ein. Werden diese Informationen in Abbildung 2.4 einbezogen, so ergibt sich der in Abbildung 2.5 dargestellte Gesamtchargenbaum.

Nun ist noch für Eingekauft_D und für Eingekauft_E zu prüfen, ob einfließende Produkte vorhanden sind. Dies ist nicht der Fall, somit ist der in Abbildung 2.5 dargestellte Chargenbaum der finale Chargenbaum für das Zielprodukt.

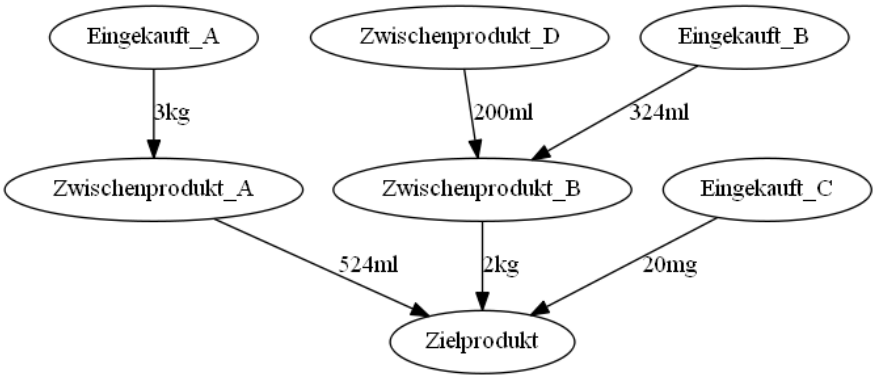

Abbildung 2.4: Chargenbaum auf Ebene zwei

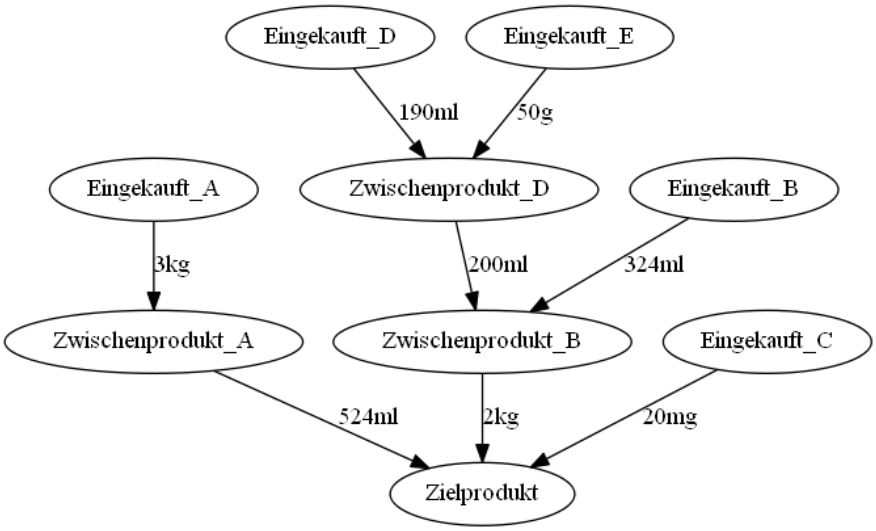

Abbildung 2.5: Vollständiger Chargenbaum auf Ebene drei

Tabelle 2.8: Beispielhafte Aktivitäten

Quelle	Ziel	Aktivitäten
Eingekauft_C	Ziel	Kontrollunterschrift: Max Mustermann
Eingekauft_E	Zwischenprodukt_D	Dosierung beachten

Chargenbaum Erweiterungen

Die grundlegende Struktur eines Chargenbaumes beschreibt lediglich die Materialflüsse im Fertigungsprozess. Dies ist oftmals nicht ausreichend, da die Umstände der Fertigung nicht erfasst werden.

Um auch Zwischenfälle oder andere Ereignisse während der Fertigung erfassen zu können, werden Chargenbäume um Aktivitäten erweitert.

Aktivitäten werden für gewöhnlich an einem Materialfluss abgelegt und können nahezu beliebige Informationen enthalten. Hierzu zählen beispielsweise Messwerte von verschiedenen Sensoren, Kontrollunterschriften oder konkretere Informationen zu den Fertigungsschritten. Diese Aufzählung ist bei weitem nicht erschöpft und soll lediglich einige Beispiele liefern.

Im Folgenden wird kurz erläutert, wie der in Tabelle 2.7 abgelegte Chargenbaum erweitert um die in Tabelle 2.8 dargestellten Aktivitäten aussieht. Für dieses Beispiel werden Aktivitäten als beliebiger Text angesehen, um das Konzept verständlich zu erläutern.

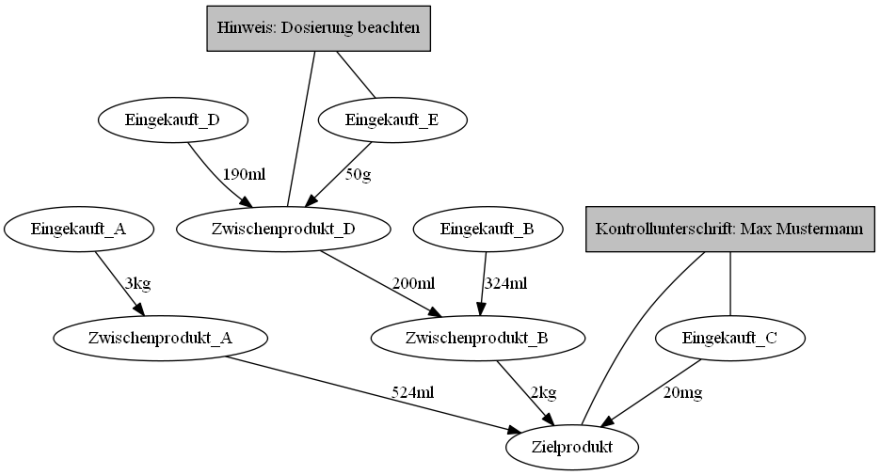

Abbildung 2.6: Vollständiger Chargenbaum um Aktivitäten erweitert

Charakterisierung

Im Allgemeinen lassen sich Chargenbäume ohne Wissen über den Prozess oder Beispieldaten, welche sie beschreiben, nicht charakterisieren. Typischerweise weisen sie jedoch nicht weniger als 100 Elemente in ihrer DAG-Repräsentation auf. Auch die Anzahl von Aktivitäten lässt sich nur schwer schätzen, da sowohl ihre Anzahl als auch die Art der Aktivitäten stark vom Prozess abhängig sind. Es können beliebige Aktivitäten mit einem beliebigen Inhalt definiert werden, was ihre Auswertung erschwert. Eine Charakterisierung der konkreten vorliegenden Daten wird in Abschnitt 3.3 durchgeführt.

2.4 Aufgabenstellung

In der pharmazeutischen Herstellung von Produkten geht es oftmals um große Geldsummen. Gerade bei biotechnischen Prozessen, wo beispielsweise eine sehr zeitaufwendige Vergärung ins Spiel kommen kann.

In der Praxis werden Fertigungsprozesse in der Pharmaindustrie aufgrund von Regulierungen der amerikanischen Food and Drug Administration (FDA) sehr ausgiebig dokumentiert, der gesamte Fertigungsprozess muss zu einem späteren Zeitpunkt nachvollzogen werden können. Es muss grundsätzlich jeder Arbeitsschritt innerhalb der Fertigung dokumentiert werden.

Es existiert als eine Datenbasis von Complex Objects und diese wiederum haben in ihren konkreten Instanzen konkrete Annotationen, welche beschreiben, ob es sich um eine gute oder schlechte Charge handelt. Ziel ist es, eben diese Annotationen rein anhand des Chargenbaumes automatisch zu generieren, um zu prüfen inwieweit rein auf Basis des Fertigungsprozesses die Qualität ermittelt werden kann, also ob es dokumentierte Abweichungen im Prozess gibt, welche für die Qualitätsschwankungen verantwortlich sind.

Diese Prozessdokumentationen werden eine gewisse Zeit archiviert und müssen aus rechtlichen Gründen verfügbar gehalten werden.

Hierdurch fallen verschiedenste Formen von Logfiles in großen Mengen an, welche wie beschrieben unter anderem auch Complex Objects enthalten. Eben diese Complex Objects enthaltenden Logfiles sollen in der Arbeit auf ihre Eignung für eine Analyse anhand des KDD Prozess untersucht werden. Das konkrete Ziel ist es hierbei, zwischen überdurchschnittlich guten oder unterdurchschnittlich schlechten Fertigungsläufen zu unterscheiden. Als Qualitätsmaß kommt der Ertrag des Prozesses in Frage. Es handelt sich also um ein Klassifikationsproblem, jedoch wäre statt dieser Betrachtung des Problems auch die Betrachtung als Regressionsproblem möglich. Anhand des Ertrages lässt sich das Qualitätsmaß automatisch generieren. Da der Durchschnitt als Messlatte angenommen wird, sind immer eine gewisse Menge schlechte

Fertigungsläufe vorhanden. Dies wurde bewusst so gewählt, da der KDD Prozess bzw. der Klassifikator so Bestandteil eines kontinuierlichen Produktionsoptimierungs-Prozesses sein kann.

Ziel ist es dabei nicht, in dieser Arbeit einen möglichst guten Klassifikator zu entwickeln, sondern verschiedene Klassifikationsverfahren auf ihre Eignung zu prüfen und ihre Leistungsfähigkeit auf den Daten zu vergleichen, um mögliche weitere Ansätze für eine spätere Verfeinerung des Mining Prozesses offen zu legen.

Bei den für die Analyse vorgesehenen Daten handelt es sich um unbereinigte Realdaten, welche einen einzelnen Fertigungsprozess mit einer Durchlaufdauer von mindestens 14 Tagen beschreiben. Um mit diesen Daten arbeiten zu können, ist einiges an Bereinigung sowie Datentransformation notwendig, um die Daten in ein Format zu bringen, auf welches maschinelle Lernverfahren (ML Verfahren) angewendet werden können. In diesem Zuge wird ebenfalls der KDD Prozess evaluiert und die Eignung für die Bearbeitung von Complex Objects soll erprobt werden. Insbesondere soll erprobt werden, inwieweit es möglich ist, die Prozessdaten, welche aufgrund ihrer internen Struktur und Komplexität für klassische Analysen eher schwer zugänglich sind, unter Anwendung der Schritte des KDD Prozesses für Analysen zugänglich zu machen.

Es stehen jedoch lediglich Daten für 126 Durchläufe über einen Zeitraum von 51 Monaten zur Verfügung. Dies erschwert das Arbeiten mit Modellen, wie z. B. mit neuronalen Netzen, die von einer großen Datenbasis profitieren.

Aus diesem Grund werden zwei Ansätze verfolgt. Einerseits werden erklärbare ML Verfahren, welche potentiell Ursachen aufzeigen können, und oftmals mit einem kleinerem Datenbestand auskommen, angewendet. Andererseits erfolgt auch die Betrachtung von neuronalen Netzen, welche aufgrund ihrer Natur zu den nicht erklärbaren ML Verfahren gehören. Ziel dieser unterschiedlichen Ansätze ist eine potenzielle frühzeitige Erkennung von Durchläufen minderwertiger Qualität, um die Fertigung ggf. vorzeitig abbrechen zu können oder eine gezielte manuelle Kontrolle oder gar das Aufdecken von Ursachen für schlechte Qualität in der Fertigung zu ermöglichen.

3 Praktische Experimente

Aufbauend auf dem Kapitel der vorbereitenden Analyse Abschnitt 2 wird in diesem Abschnitt der Arbeit die praktische Durchführung des KDD Prozesses betrachtet und entsprechend dokumentiert. Hierzu werden die einzelnen Schritte jeweils durchlaufen und Ergebnisse, beziehungsweise Probleme, und deren Lösungen oder potenzielle Lösungsansätze dokumentiert. Auch erfolgt am Ende jedes Abschnittes dieses Kapitels eine Darstellung des KDD Prozesses, in welchem der aktuelle Fortschritt sowie die während der Bearbeitung angefallenen Artefakte einbezogen werden.

3.1 Verwendete Software-Toolchain

Die im Rahmen dieser Arbeit durchzuführenden Experimente wurden in den unterschiedlichen Schritten des KDD Prozesses unter Verwendung unterschiedlicher Software Lösungen durchgeführt. In der Abbildung 3.1 werden die verwendeten Softwareprodukte den einzelnen Schritten zugeordnet.

Für große Teile der Durchführung wird die Skriptsprache Python verwendet. Dies ist ihrer großen Flexibilität und der Fülle an verfügbaren Bibliotheken geschuldet. Die Datenhaltung wird über eine MySQL Datenbank realisiert. Zum erleichterten Arbeiten wird das Werkzeug MySQL Workbench verwendet. Ergänzend zu Python wird im Rahmen der Daten Transformation die freie Software Knime eingesetzt, bei welcher es sich um eine Arbeitsumgebung zur interaktiven Datenanalyse handelt [BCD+09].

Ergänzend zu Knime kommt ebenfalls das Python Framework Tensorflow [AAB+15] zum Einsatz, welches insbesondere für das Arbeiten mit neuronalen Netzen gut geeignet ist.

3.2 Datenselektion

Der Schritt der Datenselektion des KDD Prozesses wurde durch einen Mitarbeiter der Firma, welche die Daten bereitstellt, durchgeführt, sodass darauf keinerlei Einfluss besteht. Die Daten wurden in Form einer xlsx (Microsoft Excel 2010), einer xls (Microsoft Excel 1997) und einer csv Datei bereitgestellt, welche allesamt unterschiedliche und sich ergänzende Daten enthalten.

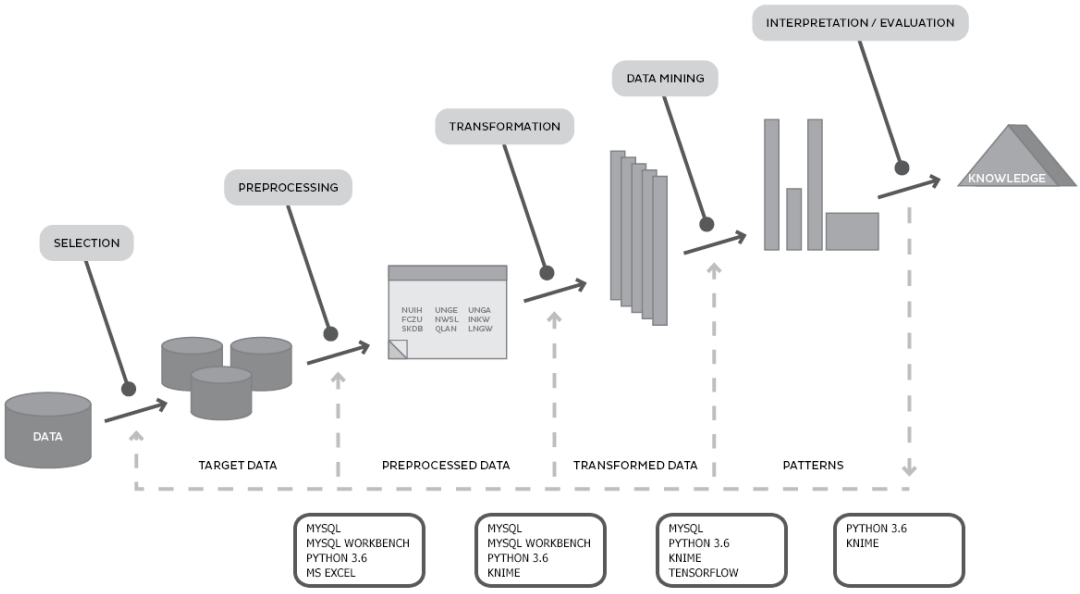

Abbildung 3.1: KDD-Prozess - Software Übersicht

Jeweils eine der Dateien repräsentiert eine Tabelle einer relationalen Datenbank, jedoch wurden nicht alle Daten übermittelt und hier bereits eine Filterung seitens des Zulieferers durchgeführt. Inwieweit relevante Daten durch diese Filterung verborgen werden, muss im Folgenden in der Analyse der Datenbasis geprüft werden.

Um mit diesen Daten arbeiten zu können, wurde im ersten Schritt eine Überführung in eine MySQL-Datenbank und eine Analyse der Datenbasis durchgeführt. Diese Analyseschritte werden im nachstehenden Kapitel genauer beleuchtet.

Stand des Prozesses

In der Abbildung 3.2 sind die aktuell generierten Artefakte des KDD Prozesses hervorgehoben. Es liegen nach der Selektion Zieldaten (Target Data) in Form vor mehreren separaten Tabellendateien vor. Diese Daten wurden bis dato keinerlei Vorverarbeitungsmechanismen im Rahmen dieser Arbeit unterzogen. Es handelt sich also um die unverarbeiteten Rohdaten, auch wenn diese teilweise bereits gefiltert wurden, welche in den folgenden Schritten umgewandelt werden, um das Ziel des KDD Prozesses zu erreichen.

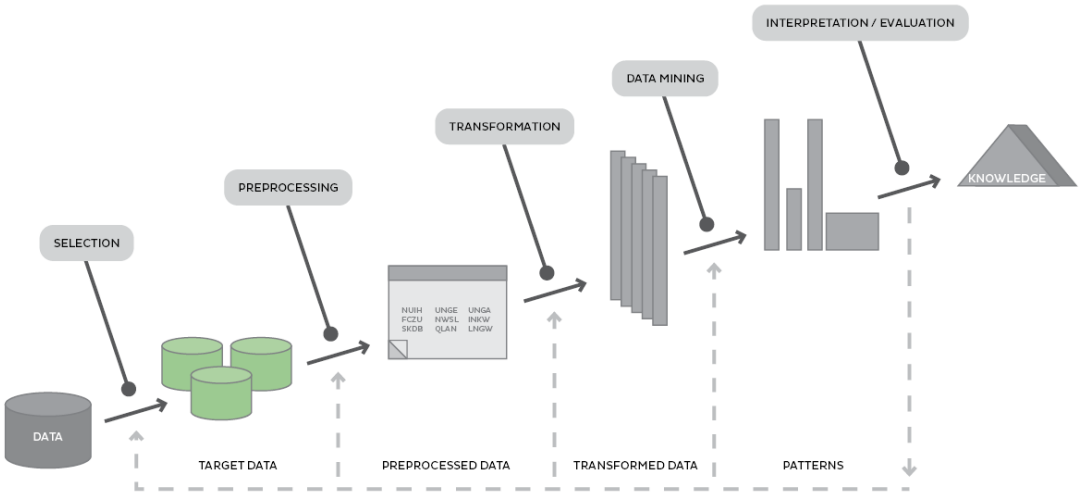

Abbildung 3.2: KDD Prozess - Target Data

3.3 Analyse der Datenbasis

Im folgenden Abschnitt wird sowohl die Struktur der Datenbasis als auch der Zustand dieser analysiert. Bei der Ermittlung des Zustandes der Datenbasis geht es hauptsächlich um verschieden ausgeprägte fehlerhafte Daten. Konkrete Konzepte für den Umgang mit eben solchen Daten werden an dieser Stelle noch nicht beleuchtet, eine Ausführung findet sich in Abschnitt 3.4.

Eine Analyse der Daten ist im KDD Prozess zwar nicht explizit erwähnt, sie ist in den Schritt der Selektion integriert. Da jedoch die Selektion durch ein externes Unternehmen durchgeführt wurde, wird dieser Schritt hier nochmals betrachtet.

3.3.1 Struktur der Datenbasis

Die Zulieferung der Daten erfolgte in ursprünglich drei Dateien mit je unterschiedlichen Formaten. Jede dieser Dateien enthält jeweils eine Tabelle eines Datenbankmodells, welche, wenn zusammengeführt, die Datenbasis bilden.

Die drei Tabellen heißen „aktivitaeten“, „Baum_Kern“ und „FertigungsZiel“, im Folgenden wird die Struktur der Tabellen dargestellt, und die Inhalte der einzelnen Tabellen, sowie das Gesamtbild werden jeweils aufgeschlüsselt. Um diese Informationen zu erlangen, wurden firmeninterne Interviews mit Experten für diese Datenstruktur und die Chargenbaum-Problematik geführt.

Tabelle 3.1: Struktur der „Baum_Kern“ Tabelle

ID	FerAufNumFlag	QuellMatBez	QuellMatNum	QuellChaNum	QuellFerAufNum	QuellMenge	QuellMengeEinheit	QuellLauNum	ZielMatNum	ZielCharNummer	ZielFertAufNum	ZielLauNum
1	Normal	MatA	1	981	1	91	kg	1	3	981	3475	1
2	Normal	MatB	2	982	1	0.2	kg	1	3	981	3475	1
3	Normal	MatC	3	983	3475	3.8	kg	248	6	986	1029	976
4	Normal	MatD	4	984	4475	13.8	kg	248	6	986	1029	976
5	Reinig	MatE	5	985	1279	6.3	kg	48	21	789	1000	712
6	Normal	MatF	6	986	1029	0.75	kg	976	8	988	2793	249
7	Normal	MatG	7	987	12	80.5	kg	1	8	988	2793	249
8	Normal	MatH	8	980	13	0.25	kg	1	9	988	2793	249
9	Normal	MatI	9	988	2793	0.2	kg	1	10	212	2310	614

Des Weiteren erfolgte noch die Nachlieferung einer vierten Datei, welche zusätzliche Daten der „aktivitaeten“ Tabelle enthält. Jedoch handelte es sich bei der Datei um eine defekte CSV Datei. In den Daten sind eine Vielzahl von Texten enthalten, welche an unterschiedlichen Stellen die in der CSV verwendeten Trennzeichen enthalten, sodass ein automatisiertes Auslesen der CSV Datei nicht möglich ist. Aufgrund der Größe der Datei von über 100mb wurde von einer händischen Bereinigung in dieser Arbeit abgesehen.

Eine konkrete Beschreibung der Attribute aller Tabellen findet sich im Anhang dieser Arbeit.

Baum_Kern

Diese Tabelle enthält die Grundstruktur der Chargenbäume und wird durch die anderen Tabellen um Informationen bereichert. Bei den Daten, welche in der Tabelle 3.1 gegeben sind, handelt es sich um abgewandelte Originaldaten, um anhand eines kleinen Beispieles einen vollständigen Chargenbaum vorstellen zu können. Die Struktur der Tabelle ist von dieser Abwandlung hierbei nicht betroffen.

Der Inhalt der Tabelle beschreibt den Fluss von Materialien in einem Fertigungsprozess. In einem Datensatz wird abgelegt, wie viel von welchem konkretem Material, über Chargen und Materialnummer identifiziert, in welches andere konkrete Material eingeflossen ist.

Somit ist es nun möglich, über die Quell- bzw. Zielchargennummer, in Kombination mit der Materialnummer, einen entsprechenden Graphen aufzubauen. Über solch ein Vorgehen lässt sich aus den in der Tabelle gegebenen Daten die Abbildung 3.3 aufbauen.

Hierbei fällt auf, dass wir nur die Materialflüsse betrachten können, welcher auch in der Tabelle erfasst sind. Informationen wie etwa Ursprung eines Ausgangsmaterials oder Verwendung des Zielmaterials sind nicht bekannt. Rückschlüsse auf einen Zulieferer sind somit im Allgemeinen nicht möglich, da das Material auch im Rahmen einer anderen internen Fertigung entstanden sein könnte, über die keine Informationen vorliegen.

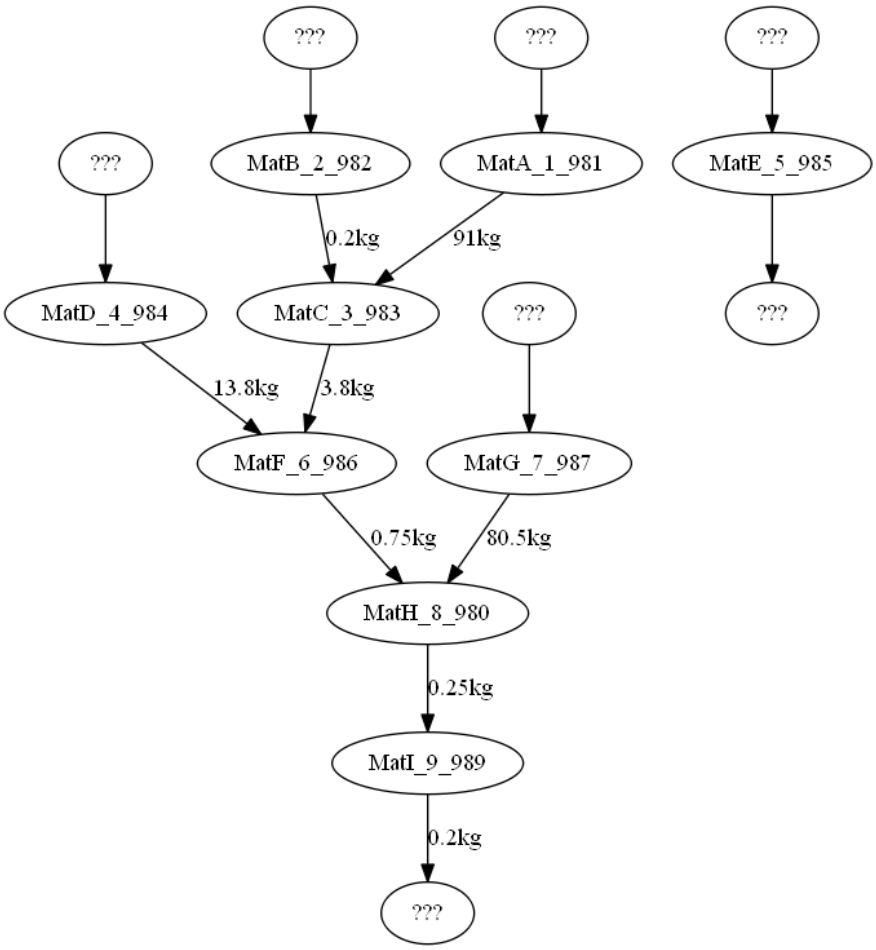

Abbildung 3.3: Aus Tabelle 3.1 generierter Graph

Tabelle 3.2: Struktur der „FertigungsZiel“ Tabelle

ID	AARF_LauNum	AARF_FerAufNum	AERI_FurChaNum	AARF_MatNum	AERI_FloWer	IstEin_AERI_Dat
1	614	2310	212	10	10.8	26.09.2012
2	598	8195	19089	198	10.24	03.12.2012
3	599	8237	19071	198	7.87	06.12.2012
4	601	56	11778	198	12.57	04.06.2013
5	602	7607	11912	198	10.86	06.06.2013
6	603	153	11779	198	11.77	10.06.2013
7	604	195	11781	198	10.41	12.06.2013
8	605	9662	11938	198	10.95	17.06.2013
9	1205	23463	10090	198	4.76	25.09.2013

FertigungsZiel

Die „FertigungsZiel“ Tabelle erweitert die „Baum_Kern“ Tabelle um Informationen über die Endprodukte eines Prozesses. Aus ihr lässt sich ebenfalls ein Qualitätsmaß herleiten, welches als Vorhersageziel des Data Mining Prozesses betrachtet wird. Ihre Struktur lässt sich der Tabelle 3.2 entnehmen.

Der Inhalt der Tabelle 3.2 enthält lediglich die Endprodukte aller Fertigungsläufe, angereichert um einen Qualitätsmesswert. Da sie für jedes Zielmaterial einen Eintrag aufweist, kann sie genutzt werden, um die in der „Baum_Kern“ Tabelle enthaltenen Daten in jeweils einzelne Fertigungsläufe zu zerlegen. Zu beachten ist allerdings, dass sich Chargenbäume auch teilweise überschneiden können.

Unter Anwendung der „target“ Tabelle lässt sich der Graph, welcher aus der „Baum_Kern“ Tabelle generiert wurde, um eine Senke mit definierter Qualität erweitern, es ergibt sich somit Abbildung 3.4. Über das dadurch eingeführte Qualitätsmaß ist es möglich, Lernverfahren anzuwenden.

Aktivitäten

Um den Fertigungsprozess, welcher in der „Baum_Kern“ Tabelle beschrieben wird, um Ereignisse und Informationen anzureichern, wird die „aktivitaeten“ Tabelle verwendet. In dieser Tabelle werden zu einem Eintrag der „Baum_Kern“ Tabelle jeweils beliebig viele Informationen verschiedenster Art abgelegt, beispielsweise können hier Kontrollunterschriften, Fehlerfälle oder Messergebnisse abgelegt werden. Aufgrund der großen Varianz der Art von Daten, welche in dieser Tabelle abgelegt werden können, ist die Struktur der Tabelle recht umfangreich. Eine tabellarische Darstellung mit einem beispielhaften Datensatz ist in Tabelle 3.3 abgebildet.

Grundsätzlich erfolgt die Zuordnung einer Aktivität zu einem Eintrag in der „Baum_Kern“ Tabelle über die Chargen und die Materialnummer des Quellmaterials, jedoch werden auch

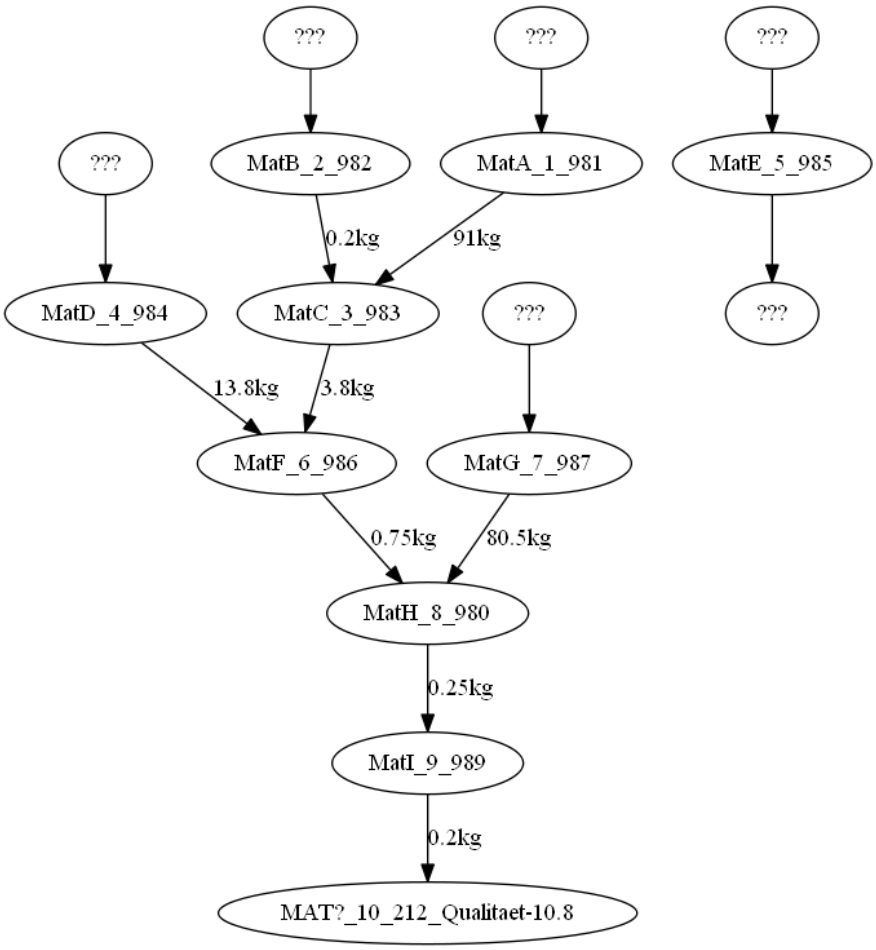

Abbildung 3.4: Um FertigungsZiel erweiterter Graph

Tabelle 3.3: Struktur der „aktivitaeten“ Tabelle

Feldname	Beispielwert
ID	12567
AKRA_AktNum	AktiNR1
AKRA_AktBez	Gewicht messen
AKRA_AktTyp	Manuelle Aktivität
AWRA_ArbAnwNum	AnweisungNr5
AWRA_ArbAnwBez	Die Bezeichnung, oft kryptischer Wert
AWRA_Typ	Manuell
AARF_LauNum	LaufNr10
AARF_FerAufNum	FertigungsAuftragsNr3
AERI_FurChaNum	ChargeNR234
AARF_MatNum	Mat66
AARF_MatBez	Kaliumhexacyanoferat
AARF_HerVorVar	VarianteA
AARF_HerVorVer	Verion1
AARF_IstSta	2017-06-22 18:00
AARF_IstEnd	2017-06-22 18:10
AARP_ProEin	WagenraumA
AGRA_ArbGanNum	GangNr16
AGRA_ArbGanBez	Wiegen
AERS_SolBooWer	???
AERI_FloWer	20.0
AERI_BooWerTex	???
AERI_TexWer	20kg
AERI_Tex	???
AERI_DATUM	2017-06-22 18:10

weitere Informationen wie etwa die Laufnummer oder die Fertigungsauftragsnummer aus der „Baum_Kern“ Tabelle übernommen und redundant abgelegt. Die konkrete Bedeutung der einzelnen Felder ist stark von den zu speichernden Daten abhängig, sodass ihnen nicht immer eine eindeutige Bedeutung zugeordnet werden kann.

3.3.2 Überführung in eine relationale Datenbank und Probleme

Um mit den Daten, welche in den drei verwendbaren Dateien vorliegen, arbeiten zu können und sie für einen automatisierten Zugriff besser zugänglich zu machen, wurden die Daten in eine klassische relationale Datenbank überführt. Für diese Aufgabe ist grundsätzlich jede relationale Datenbankimplementation geeignet. Aufgrund der doch eher kleinen Menge von Daten und des Lizenz-Modells wird hier eine MySql Datenbank verwendet.

Die Größe der einzelnen Dateien macht einen Import über eine CSV Dateien unpraktikabel, da die Umwandlung der Excel Datei in eine CSV Datei nicht gut funktioniert. Es sind vermehrt Programmabstürze von Microsoft Excel zu verzeichnen.

Des Weiteren ist die CSV-Import Funktionalität der verwendeten Software „mysql workbench“ sehr langsam und es treten Server Timeouts auf. Die Timeouts ließen sich zwar über eine Änderung der MySQL Server Konfiguration beheben, doch ist diese Methode ohnehin sehr zeitintensiv, da erstens der Import der CSV Dateinen in die Datenbank sehr viel Zeit in Anspruch nimmt, andererseits allerdings das Erzeugen der CSV Dateien aus Excel Dokumenten ab einer gewissen Größe der Datei anscheinend sehr problematisch ist.

Um dieses Problem des Exportes zu umgehen, werden zwei andere Ansätze verfolgt, mittels Python wird ein Skript erstellt, welches die Excel bzw. vorhandenen CSV Dateien in SQL Skripte übersetzt, welche dann direkt auf der Datenbank ausgeführt werden konnten. Dieser Ansatz ist zwar erfolgreich, jedoch nicht sonderlich schnell. Die kürzeste Laufzeit aller Importansätze wird mit einem Python Skript erreicht, welches das generierte SQL Skript während der Erzeugung direkt auf der Datenbank ausführt.

Wie sich jedoch herausstellte, erfolgte die Überführung in die Datenbank unter der falschen Prämisse, dass es sich bei sogenannten Chargennummern um numerische Werte handelt. Diese Annahme führte dazu, dass acht Datensätze falsch abgelegt wurde, da sie eine Chargennummer der Struktur „xxxx-y“ aufwiesen, sie wurden als „xxxx“ in die Datenbank übernommen. Hierdurch entstand ein verfälschter Eindruck, so wiesen die Chargenbäume etwa Schleifen auf, was auf das Fehlen des „-y“ zurückzuführen ist. Glücklicherweise waren nur 16 Datensätze betroffen, sodass eine händische Reparatur der Daten in recht kurzer Zeit durchgeführt werden konnte. In diesem Zuge wurden die Daten mittels automatisiertem Vorgehen auf weitere Abweichungen von den Quelldaten untersucht und für fehlerfrei befunden.

3.3.3 Zustand der Datenbasis

Bereits eine grobe Analyse der bereitgestellten Daten zeigt, dass die Datenbank bereits die zweite Normalform verletzt und somit einiges an Potenzial für Fehler bzw. Probleme, welche einer Analyse im Weg stehen, vorhanden ist. Um nun konkrete Probleme in den Daten aufzudecken, werden sowohl manuelle als auch maschinelle Ansätze kombiniert. Hierbei werden Python Skripte und händische SQL Abfragen verwendet. Die Ergebnisse dieser Analysen werden in diesem Abschnitt der Arbeit dargelegt.

Anzahl von Einträgen in den Tabellen

Im ersten Schritt wird geprüft, wie viele Datensätze für eine Analyse vorliegen. Dabei ergibt sich, dass lediglich 126 Durchläufe vorhanden sind, jedoch insgesamt 42824 Einträge in der „Baum_Kern“ Tabelle. Das macht durchschnittlich 339.87 Chargenbaum Kanten für jeden Durchlauf, wenn man davon ausgeht, dass kein Baum_Kern Eintrag für mehrere Fertigungsläufe genutzt wird. Dazu kommen noch insgesamt 472932 dokumentierte Aktivitäten, der Durchschnitt pro Durchlauf liegt somit bei 3753.42 Aktivitäten pro Durchlauf. Zu beachten ist jedoch, dass ein Großteil der potentiell zu Verfügung stehenden Aktivitäten aufgrund der defekten CSV Datei nicht vorliegen. Listing 3.1 enthält die Querrys, welche genutzt wurden.

Listing 3.1: SQL Querry zur Bestimmung der Anzahl von Einträgen

```
1 SELECT count(*) FROM FertigungsZiel;          -- ergibt:     126
2 SELECT count(*) FROM aktivitaeten;      -- ergibt: 472,932
3 SELECT count(*) FROM Baum_Kern;  -- ergibt:  42,824
```

Diese Zahlen betreffen die gesamt vorliegenden Daten, jedoch kann die Zahl der verwendbaren Daten abweichen, da beispielsweise in der „Baum_Kern“ Chargenbäume oder Teile von Chargenbäumen vorhanden sein können, welche nicht über einen Eintrag der „FertigungsZiel“ Tabelle referenziert werden oder auch die „FertigungsZiel“ Tabelle auf nicht vorhandene oder beschädigte Chargenbäume referenziert. Im weiteren Verlauf dieses Kapitels wird geprüft, inwieweit die Daten von derartigen Problematiken betroffen sind.

Eindeutigkeit der Chargennummer

Laut Expertenaussage handelt es sich bei der Chargennummer um einen Wert, welcher nur in Kombination mit der Materialnummer zur eindeutigen Identifizierung eines konkreten Materials genutzt werden kann. Jedoch ist die Chargennummer in den hier vorliegenden Daten auch ohne eine Kombination mit der Materialnummer ein eindeutiger Wert, welcher für die Identifizierung von konkreten Materialien genutzt werden kann.

Diese Erkenntnis wurde durch die in Listing 3.2 dargestellte SQL Anfrage und einige entsprechend umgestellte Anfragen erlangt.

Listing 3.2: SQL Querry zur Ermittlung der Eindeutigkeit von Chargennummern

```
SELECT t2.chargennummer
FROM(
        SELECT t.QuellChaNum AS chargennummer,
                COUNT(*) AS c
        FROM(
                SELECT QuellChaNum,
                        QuellMatNum
                FROM Baum_Kern
                GROUP BY QuellChaNum
        ) t
        GROUP BY t.QuellChaNum
) t2 WHERE t2.c > 1;
```

Duplikate Fertigungsläufe

Bei oberflächlicher Betrachtung fällt auf, dass einige Einträge in der FertigungsZiel Tabelle doppelt auftauchen. Dies wurde bei augenscheinlicher Betrachtung für einen Fertigungslauf festgestellt. Wie die in Listing 3.3 SQL Anfrage zeigt, sind insgesamt zwei Durchläufe hiervon betroffen und liegen jeweils doppelt vor. Es sind also 1.59% der Fertigungsläufe betroffen.

Listing 3.3: SQL Querry zur Erkennung mehrfacher Einträge

```
select t.AERI_FurChaNum,t.AARF_MatNum,i from
   (select count(*) as i,
           AERI_FurChaNum,
           AARF_MatNum
      from FertigungsZiel
         group by AERI_FurChaNum,
                  AARF_MatNum
   ) t where t.i > 1;
        -- Alle Duplikaten Chargennummern in der FertigungsZiel Tabelle
```

Leere Chargenbäume

Ein weiterer Defekt in den Daten, welcher recht problematisch ist, ist die Tatsache, dass für eine gewisse Anzahl von Fertigungsläufen kein Chargenbaum vorhanden ist.

Listing 3.4: SQL Querry zur Erkennung nicht vorhandener Chargenbäume

```
select AERI_FurChaNum,
       AARF_MatNum
   from FertigungsZiel where
      AERI_FurChaNum not in
         (select cb.ZielCharNummer from
            Baum_Kern cb where
               cb.ZielMatNum = AARF_MatNum
         ); -- Alle Produkte Ohne Chargenbaum
```

Von dieser Problematik sind 15 Produktionsläufe betroffen, wie die Ergebnisse von Listing 3.4 zeigen. Somit sind für 11.90% der Produktionsläufe keinerlei Chargenbäume vorhanden, was doch ein beachtlicher Anteil ist.

Für diese Problematik kommen mehrere Ursachen in Betracht, die wahrscheinlichste Ursache ist, dass bei dem Export der Daten von Seiten des Kunden Fehler begangen wurden, und die durch den entsprechenden Eintrag der „FertigungsZiel“ Tabelle referenzierten Einträge der „Baum_Kern“ Tabelle nicht korrekt ausgewählt wurden.

Durch diesen Fehler stellt sich allerdings die Frage, inwieweit die vorhandenen Chargenbäume vollständig sind. Dies kann jedoch nicht verlässlich geprüft werden, da dies rein aus den Daten nicht entnommen werden kann. An dieser Stelle ist lediglich eine Plausibilitätsüberprüfung, etwa über die Größe eines Chargenbaumes, möglich. Dieser Ansatz wird unter Abschnitt 3.3.3 weiter verfolgt.

Kurze Chargenbäume

Die durchschnittliche Größe der Menge an in einen Chargenbaum vorhandenen Kanten, in Zukunft als Länge bezeichnet, liegt bei 208.45. Sollte ein Chargenbaum zwar vorhanden sein, jedoch weniger als 20% dieses Durchschnittswertes als Länge aufweisen, so ist er kurz, und für Analysen nur bedingt geeignet. Ursachen hierfür könnten etwa Fehler beim Export oder Fehler in der Protokollierung sein.

Zur Identifizierung dieser Chargenbäume wird ein Python Skript verwendet. Die Ergebnisse werden in der Abbildung 3.5 dargestellt.

57 Produktionsläufe weisen einen Chargenbaum mit einer Länge von weniger als 41 auf. Zieht man entsprechend die Chargenbäume ab, welche eine Länge von 0 aufweisen, so erhält man 42 Chargenbäume, die eine Länge >0, jedoch kleiner 41 haben. Dies entspricht 33.33% aller Chargenbäume. Wie im Plot der Chargenbaumlänge zu erkennen ist, scheinen fehlende Chargenbaum Informationen für Durchläufe in zeitlicher Nähe aufzutreten. Auch fällt auf,

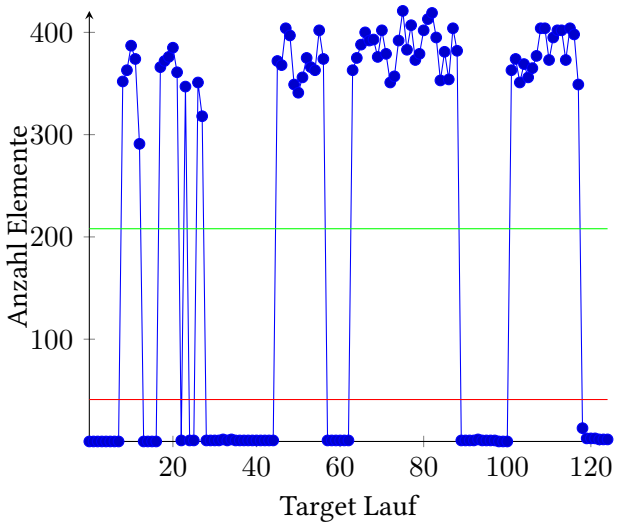

Abbildung 3.5: Länge von Chargenbäumen

dass die Längen stark polarisiert sind, die Extremen sind ziemlich stark ausgeprägt, und es liegt keine Normalverteilung vor. Werden alle Chargenbäume, welche eine Länge von weniger als 41 aufweisen, nicht bei der Durchschnittsberechnung berücksichtigt, ergibt sich für die verbleibenden 69 Chargenbäume eine durchschnittliche Länge von 376.51.

Schleifen in Chargenbäumen

Durch den Fehler, dass einige Materialnummern nicht korrekt in der Datenbank abgelegt wurden, hatte es den Anschein, als trete in 68 Chargenbäumen eine Schleife auf. Das wären 53,97% aller Chargenbäume gewesen. Aufgrund dieser unrealistischen Zahlen wurde eine Überprüfung der importierten Datenbasis angeregt, wobei der Importfehler entdeckt wurde.

Nach Korrektur des Importfehlers wurden alle Schleifen aus den aktiven Chargenbäumen entfernt.

In der „Baum_Kern“ Tabelle sind jedoch immer noch Einträge vorhanden, welche eine Schleife bilden. Jedoch werden diese durch keines der konkreten Materialien aus der „FertigungsZiel“ Tabelle, weder direkt noch indirekt, referenziert.

Transfer Mengen von Null

Ein weiteres Problem tritt in der „Baum_Kern“ Tabelle auf, in welcher die Baumstruktur enthalten ist. Rund 2.73% der Einträge in dieser Tabelle weisen eine Menge von Null auf, wie die triviale Anfrage aus Listing 3.5 zeigt. Ob es sich hierbei um einen Datendefekt handelt, ist

leider nicht klar. Es wäre auch möglich, dass etwa Behälter, welche in der Fertigung verwendet werden, auf diese Weise kodiert sind. Zur Identifizierung dieser Einträge kann eine leicht abgewandelte Form des untenstehenden Querrys verwendet werden.

Listing 3.5: SQL Querry zur Erkennung von nullwertigen Transfermengen

```
select count(1) from
   Baum_Kern where
      QuellMenge = 0;
```

Negative Transfermengen

Auch gibt es einen kleinen Prozentsatz von Einträgen in der „Baum_Kern“ Tabelle, welche eine negative Transfermenge aufweisen. Betroffen sind lediglich 0.52% aller Einträge, doch muss dringend geklärt werden, wie mit solchen negativen Transfermengen umzugehen ist. Die verwendete Anfrage findet sich in Listing 3.6 .

Listing 3.6: SQL Querry zur Erkennung negativer Transfermengen

```
select count(1) from
   Baum_Kern where
      QuellMenge < 0;
```

Baum_Kern Weisen

Es können 27347 Einträge aus „Baum_Kern“ weder einem Eintrag aus „FertigungsZiel“ noch einem anderen Eintrag aus „Baum_Kern“ zugeordnet werden. Insgesamt enthält die Tabelle 42824 Einträge, somit sind mindestens 63,86% aller „Baum_Kern“ Einträge unbrauchbar, weil sie mit keinem Fertigungslauf in Verbindung gebracht werden können, dies zeigt Listing 3.7. Im Folgenden werden Einträge ohne Referenz auf andere oder auf die „FertigungsZiel“ Tabelle als Weisen bezeichnet.

Für diese Analyse werden A und B in einem Konstrukt, wie „A->B->C“ , als Nichtweisen angenommen, auch wenn es sich bei C um eine Weise handelt, da sie auf einen Eintrag der „Baum_Kern“ Tabelle zurückgeführt werden können. Die korrekte Zahl von nicht brauchbaren Einträgen liegt somit wohl über den eben genannten 63.56%.

Listing 3.7: SQL Querry zur Erkennung von Weisen

```
select count(*) from
       (select * from
              Baum_Kern where
```

```
                        ZielCharNummer not in
                                ( select  cb.QuellChaNum from
                                        Baum_Kern cb
                                )
        ) t1 where
                t1.ZielCharNummer not in
                        (select t.AERI_FurChaNum from FertigungsZiel t)
```

3.3.4 Zusammenfassung des Zustandes

Im Allgemeinen befinden sich die Daten in einem fragwürdigen Zustand und sind aufgrund der geringen Menge von Daten und hoher Unbrauchbarkeit für ML Verfahren nur bedingt geeignet. In der Abbildung 3.6 wird die Nutzbarkeit der zur Verfügung stehenden 126 Datensätze der „FertigungsZiel“ Tabelle dargestellt. Leider ergab sich durch die Analyse, dass lediglich knapp die Hälfte der Fertigungsläufe von Problemen unbehaftet sind, es bleiben 68 unbelastete Chargenbäume.

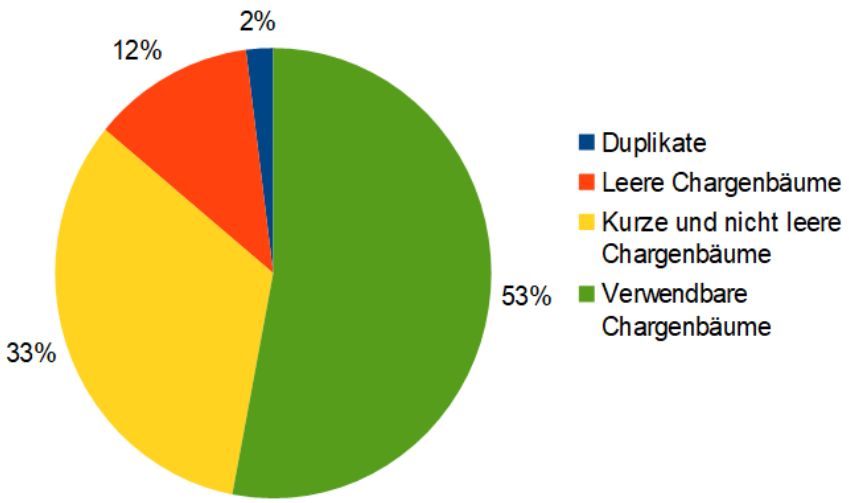

Abbildung 3.6: Brauchbarkeit der Fertigungsläufe

Während der folgenden Vorverarbeitung muss nun geklärt werden, inwieweit die Daten bereinigt werden können und sollen, um ihre Aussagekraft nicht zu verfälschen.

Tabelle 3.4: Zu entfernende Materialbezeichnungen

Typ	Feld
Text	Baum_Kern.QuellMatBez aktivitaeten.AKRA_AktBet aktivitaeten.AWRA_ArbAnwBez aktivitaeten.AARF_MatBez aktivitaeten.AGRA_ArbGanBez

3.4 Vorverarbeitung und Bereinigung

Der Schritt der Vorverarbeitung und Bereinigung befasst sich, wie in der Vorstellung des KDD Prozesses (Abschnitt 2.1.2) beschrieben, mit der Bereinigung und Aufbereitung von Problemen, welche in den Target Data vorliegen. Die konkret angewendeten Methoden und Ansätze werden hier dokumentiert.

3.4.1 Anonymisieren der Datenbasis

Da es sich bei den Daten um kundenspezifische Produktivdaten handelt, welche Rückschlüsse auf betriebliche Abläufe des Fertigungsprozesses zulassen, müssen die Daten entsprechend anonymisiert werden. Das Anonymisieren der Datenbasis erfolgt erst nach der Analyse eben dieser, um auszuschließen, dass durch den Schritt des Anonymisierens Fehler in die Datenbasis integriert werden. Die in der Analyse erlangten Erkenntnisse können nach dem Anonymisieren gegen die nun anonymisierte Datenbasis geprüft werden, um die Wahrscheinlichkeit von Fehlern zu reduzieren.

Die konkreten Maßnahmen, welche notwendig sind, um die Daten entsprechend zu anonymisieren, wurden firmenintern kommuniziert, und es ergab sich das Konzept, welches im Folgenden beschrieben wird.

Entfernung von Beschreibungstexten

Grundsätzlich müssen alle Texte, welche Rückschluss auf Vorgehen oder verwendetes Material zulassen, entfernt oder unkenntlich gemacht werden. Da jedoch immer eine Identifikationsnummer in Kombination mit dem beschreibenden Text verwendet wurde, können die Beschreibungstexte ohne Informationsverlust einfach entfernt werden. Da in der „FertigungsZiel“ Tabelle keinerlei Beschreibungstexte abgelegt werden, ist sie von dieser Maßnahme nicht betroffen. Eine Übersicht der betroffenen Felder findet sich in Tabelle 3.4.

Gerade in der „aktivitaeten“ Tabelle sind jedoch noch weitere Textfelder vorhanden, dessen Veränderung die Aussage der betroffenen Aktivitäten verfälschen würde. So ist beispielsweise für einige Aktivitäten, welche eine Kontrollunterschrift repräsentieren sollen, der Name des Unterschreibenden in Klartext hinterlegt, was aus Datenschutzgründen sehr bedenklich ist.

Anonymisieren von Produktionseinheiten

Das Feld AARP_ProEin der Aktivitäten Tabelle enthält einen Text, welcher die Produktionseinheit identifiziert, in welcher die Aktivität aufgetreten ist. Dieses Feld muss entsprechend unkenntlich gemacht werden, hierzu wurden alle Produktionseinheiten ermittelt, welche in den Daten Verwendung finden und diese jeweils durch eine Nummer einer aufsteigenden Sequenz ersetzt. Dieser Schritt wurde mit einem Python Skript durchgeführt.

Anonymisieren von Identifikationsnummern

Ähnlich wie auch bei den Produktionseinheiten besteht die Anforderung, die Identifikationsnummern von beispielsweise Material oder Aktivitäten zu anonymisieren. Hierzu wurden sämtliche numerische Attribute erfasst und in tabellarischer Form je nach enthaltendem Wertetyp gruppiert (Tabelle 3.5).

Es wurde für jede Art von Inhalt der gesamte Wertebereich ermittelt und auf den Zahlen eins bis *Anzahl Werte* abgebildet. Dieser Prozess wurde unter Verwendung einfacher SQL Querrys und einiger Python Skripte vollständig automatisiert.

Entfernung der Aktivitäten Tabelle

In der Aktivitäten Tabelle befinden sich teilweise Aufzeichnungen, welche personenbezogene Daten enthalten, wie beispielsweise Namen von Angestellten. Um alle relevanten Typen von Aktivitäten zu identifizieren, ist es notwendig, alle Arten von Aktivitäten zu betrachten. Dies ist aufgrund der riesigen Menge von Aktivitätsarten nicht ohne weiteres möglich. Dies ist einer der Gründe, warum im Folgenden Aktivitäten nicht weiter betrachtet werden. Neben dieser Problematik wird diese Entscheidung auch durch die Unvollständigkeit der Aktivitätsdaten unterstützt.

3.4.2 Maßnahmen zur Bereinigung der Datenbasis

Neben der Anonymisierung der Datenbasis ist die Behandlung der in Abschnitt 3.3.3 aufgeführten Defekte ein wichtiger Teil der Vorverarbeitung.

Tabelle 3.5: Zu anonymisierende numerische Werte

Typ	Feld
Materialnummern	Baum_Kern.QuellMatNum Baum_Kern.ZielMatNum FertigungsZiel.AARF_MatNum aktivitaeten.AARF_MatNum
Chargennummern	Baum_Kern.QuellChaNum Baum_Kern.ZielCharNummer FertigungsZiel.AERI_FurChaNum aktivitaeten.AERI_FurChaNum
Fertigungsauftragsnummern	Baum_Kern.QuellFerAufNum Baum_Kern.ZielFertAufNum FertigungsZiel.AARF_FerAufNum aktivitaeten.AARF_FerAufNum
Laufnummern	Baum_Kern.QuellLauNum Baum_Kern.ZielLauNum FertigungsZiel.AARF_LauNum aktivitaeten.AARF_LauNum
Aktivitaetennummern	aktivitaeten.AKRA_AktNum
Arbeitsanweisungsnummern	aktivitaeten.AWRA_ArbAnwNum
Arbeitsgangsnummern	aktivitaeten.AGRA_ArbGanNum
Herstellungsvorschritsvarianten-Nummern	aktivitaeten.AARF_HerVorVar

Da jedoch für die Behandlung eben dieser Defekte ein gewisses Maß an Wissen über den Fertigungsprozess und allgemeine betriebliche Abläufe notwendig ist, wurden im Rahmen eines Experteninterviews mit einem leitenden Mitarbeiter der Firma, von welcher die Daten stammen, die gefundenen Probleme besprochen und mögliche Lösungsansätze diskutiert. Diese Erkenntnisse fließen maßgeblich in die folgenden Unterkapitel ein. Jedoch lief es im Endeffekt auf ein Ignorieren der meisten defekten Daten hinaus.

Duplizierte Fertigungsläufe

Die duplizierten Einträger aus der „FertigungsZiel“ Tabelle sollen entfernt werden, sodass jeweils lediglich ein Eintrag vorliegt. Hierdurch wird das Problem für die Analyse umgangen, jedoch sollte geprüft werden, aus welchen Gründen das mehrfache Auftauchen eines Fertigungslaufes auftreten konnte. Durch diese Vorgehen wird der Samplespace von 126 auf 124 Einträge reduziert. Konkret sind Einträge 109, 110 sowie 113, 114 jeweils identisch. Anschließend wurden Einträge mit jeweils der höheren ID entfernt.

Leere Chargenbäume

Im Zuge der Bereinigung werden leere Chargenbäume aus der Datenbasis entfernt, wodurch die Anzahl der insgesamt verwendbaren Datensätze weiter von 124 auf 109 fällt. Ebenso wie bei duplizierten Einträgen können fehlende Chargenbäume auf schwerwiegende Konsistenzprobleme in der Quelldatenbank hindeuten. Auch wenn es wahrscheinlich ist, dass das Fehlen der Einträge auf einen Fehler bei der ursprünglichen Datenselektion hindeutet, sollte die Quelldatenbank auf ihre Konsistenz hin untersucht werden. Das Löschen der leeren „FertigungsZiel“ Einträge wurde mittels eines einfachen Python Skriptes durchgeführt.

Kurze Chargenbäume

Auch mit kurzen Chargenbäumen wird verfahren wie mit Leeren. Jedoch stellt sich auch hier wieder die Frage, inwieweit diese Chargenbäume durch einen Exportfehler entstanden sein können. Auch an dieser Stelle sollte die Datenbank entsprechend überprüft werden. Die 26 kurzen Chargenbäume wurden mittels eines Python Skriptes aus der „FertigungsZiel“ Tabelle entfernt, die vorhandenen Einträge in der „Baum_Kern“ Tabelle werden nicht entfernt. Dies ist unproblematisch und auch notwendig, da ohne weitere Prüfung nicht sichergestellt werden kann, dass die betroffenen core Einträge nicht auch Teil eines anderen Chargenbaumes sind.

Baum_Kern Weisen

Mit ziemlicher Sicherheit können die „Baum_Kern“ Einträge ohne Referenz auf einen „FertigungsZiel“ Eintrag auf einen Fehler während der Datenselektion zurückgeführt werden, da in dem System auch andere Produktionsläufe erfasst werden, welche anscheinend nicht vollständig herausgefiltert wurden. Von daher werden diese Datensätze in den kommenden Phasen nicht weiter betrachtet. Ein Löschen der Einträge ist nicht notwendig, da keine Referenz zu einem Eintrag der „FertigungsZiel“ Tabelle vorhanden ist und sie somit auch keinem Fertigungslauf zugeordnet sind.

Schleifen in Chargenbäumen

Durch den Fehler während des Imports der Daten aus den CSV bzw. XLS/XLSX Dateien sah es so aus, als würde die Mehrzahl aller Chargenbäume Schleifen aufweisen. Dies ist glücklicherweise nicht der Fall. Nach Korrektur des Fehlers verblieben lediglich zwei Schleifen, welche jedoch allesamt innerhalb von Chargenbaum Weisen auftreten. Daher muss an dieser Stelle kein besonderes Vorgehen zum Umgang mit diesen deklariert werden, durch das Verwerfen von Chargenbaum Weisen werden diese bereits implizit entfernt. Jedoch sollte, auch wenn nicht im Rahmen dieser Arbeit, geprüft werden, wieso überhaupt Schleifen in den Chargenbaum Daten auftreten und welche semantische Bedeutung ihnen zukommt.

Unplausible Transfermengen

Auch in Kundeninterviews konnte leider die Bedeutung von Transfermengen <=0 nicht geklärt werden. Aus diesem Grund werden sie nicht gesondert behandelt und fließen im selben Maße wie Transfermengen >0 in die kommenden Analysen ein.

Stand des Prozesses

Wie die Abbildung 3.7 zeigt, wurden die als Ergebnis der Selektion vorliegenden Target Data in Preprocessed Data überführt. Die Preprocessed Data liegen nun in Form einer MySQL Datenbank vor und müssen mittels des Transformationsschrittes des KDD Prozesses in ein Format überführt werden, mit welchem ML Verfahren betrieben werden können.

3.5 Daten Transformation / Linearisierung

Die zu betrachtenden Verfahren sind nicht direkt auf Daten einer Baumstruktur anwendbar, auch ist fraglich, wie sinnvoll eine direkte Analyse der Bäume ist, da diese teilweise doch recht

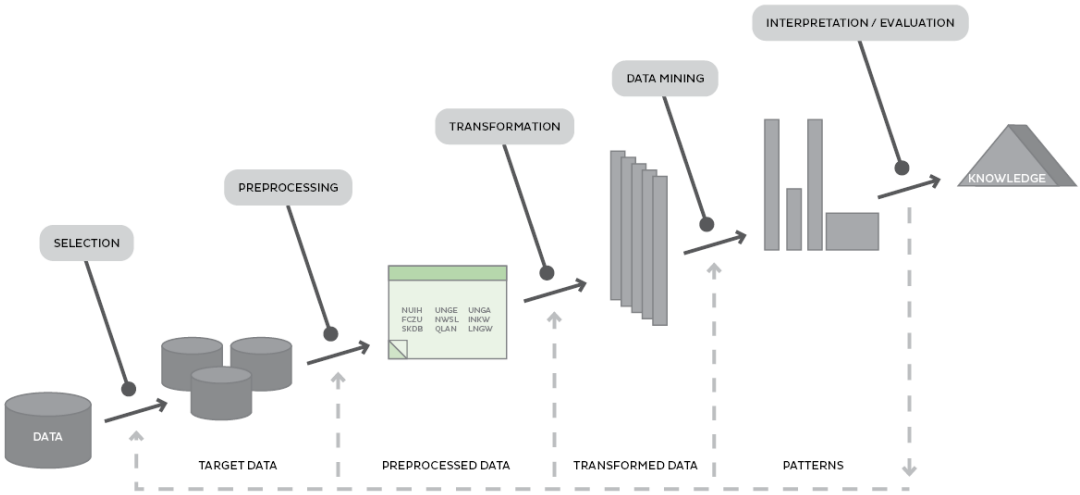

Abbildung 3.7: KDD-Prozess - Preprocessed Data

unterschiedliche Strukturen aufweisen können. Daher ist es notwendig, diese in ein anderes Format umzuwandeln, mit welchem an ihrer Stelle gearbeitet werden kann. Daher werden verschiedene Verfahren zur Linearisierung der Daten betrachtet, hierbei wird auch der Verlust von Informationen in Kauf genommen.

3.5.1 Ermittlung der einfließenden Teilprodukte

Eine Möglichkeit die Produktionsläufe in ein Datenformat zu bringen, welches für eine mechanische Analyse geeignet ist, wäre es, die in ein Endprodukt einfließenden Ausgangsmaterialien und Zwischenprodukte zu ermitteln und auf Basis der einfließenden Mengen eine Analyse durchzuführen. Es könnte nun für jeden Fertigungslauf ermittelt werden, wie viel eines welchen Materials verwendet wurde. Problematisch ist jedoch, dass es auch einen geringen Anteil von Daten gibt, welche eine Transfermenge von Null aufweisen, sodass sie in dieser Repräsentation nicht vertreten sind.

Die Aktivitäten Tabelle wird für diese Art der Transformation nicht benötigt, jedoch wäre es durchaus möglich, diese Informationen einfließen zu lassen, falls sie nicht entfernt worden wären.

Es sind verschiedene Ansätze zur Durchführung einer solchen Transformation möglich. Die Vor- und Nachteile der einzelnen Ansätze werden im Folgenden dargelegt. Hierzu wird der in Abbildung 3.8 abgebildete Baum für eine Demonstration der Verfahren verwendet.

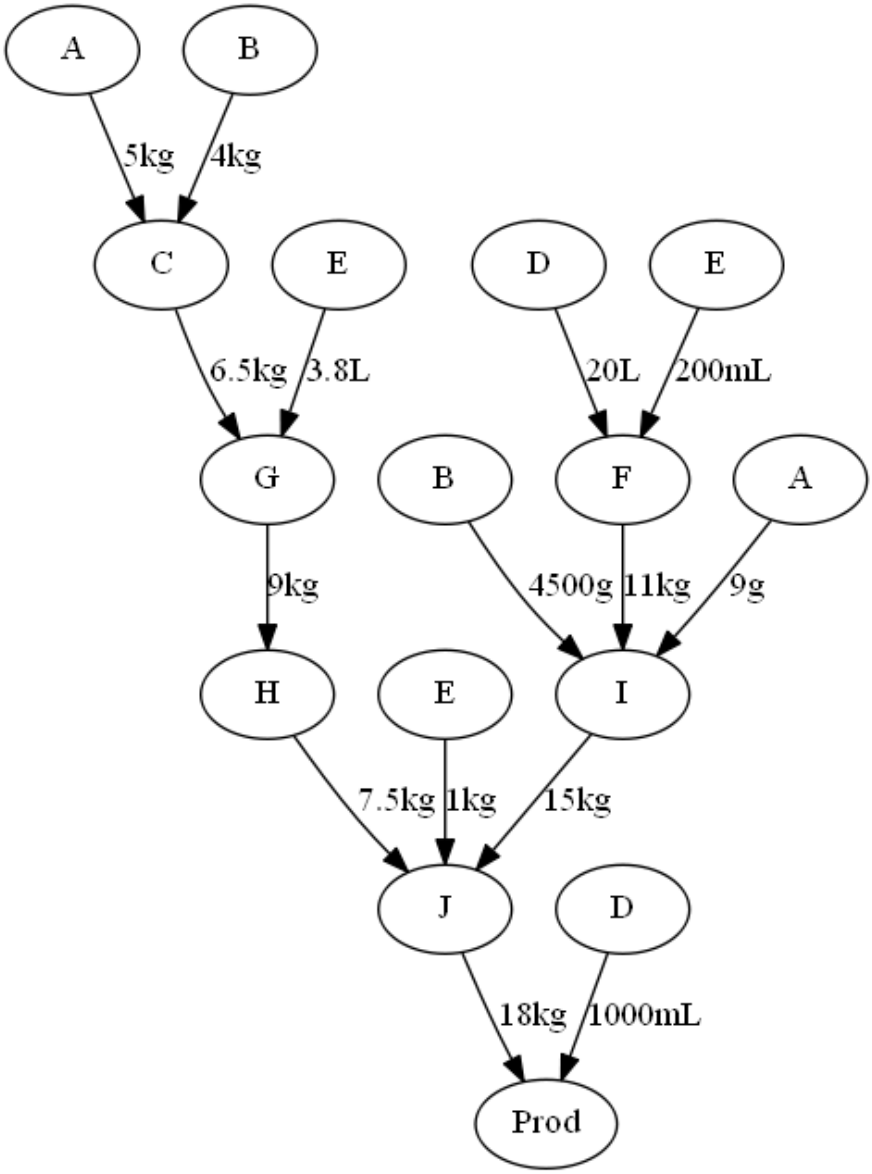

Abbildung 3.8: Einfließende Teilmengen Demo tree

Tabelle 3.6: Ermittlungen aller einfließenden Mengen naiver Ansatz

A	B	C	D	E	F	G	H	I	J
14	4504	6.5	1020	204.8	11	9	7.5	15	18

Ermittlung aller einfließenden Mengen

Ein naiver Ansatz wäre, einfach alle in eine Fertigung einfließenden Materialien zu ermitteln und die Werte, welche in den jeweiligen Transaktionen angegeben sind, ungeachtet ihrer Mengeneinheit zu verwenden und aufzusummieren.

Die hierüber erhaltenen Werte sind mit recht hoher Sicherheit nicht korrekt, da nicht immer die gesamte produzierte Menge eines Zwischenproduktes für die weitere Fertigung verwendet wird. Dies liegt daran, dass die produzierte Menge auch bei gleichen Ausgangsmengen von Grundstoffen ggf. durch nicht kontrollierbare Faktoren leicht beeinflusst wird. Somit muss grundsätzlich mehr hergestellt werden, als wirklich benötigt wird, um sicherzustellen, dass genug für die Fertigung des Endproduktes bzw. des nächsten Teilproduktes verfügbar ist. Auch werden teilweise unterschiedliche Mengeneinheiten für dasselbe Material verwendet. Diese Problematiken werden bei diesem Ansatz ignoriert, jedoch in den folgenden Ansätzen betrachtet. Die Tabelle 3.6 zeigt exemplarisch, wie dieser Ansatz aussehen könnte. Zu beachten ist, dass insbesondere bei Material A, B, D und E die errechneten Werte aufgrund der unterschiedlichen Mengeneinheiten weit von der Realität entfernt sind.

Im folgenden wird dieser Ansatz als „naiver Ansatz“ bezeichnet.

Errechnung von geschätzten einfließenden Mengen

Das Problem, dass Mengenangaben in unterschiedlichen, teilweise nicht ohne weiteres in eine gemeinsame überführbare Mengeneinheit vorliegen, lässt sich mit einer Umrechnungstabelle auf Basis von Annäherungs- oder Schätzwerten behandeln. Ziel dabei ist es, alle Mengen auf eine gemeinsame Einheit zu bringen. Dies ist insbesondere notwendig, da für das selbe Material teilweise unterschiedliche Mengeneinheiten verwendet werden, wie auch in Abbildung 3.8 für Materialien B, D und E dargestellt, und somit ein einfaches Aufsummieren ohne Umrechnungsfaktor fehlerhafte Werte liefert.

Hierfür wird eine Zieleinheit definiert und für jede Mengeneinheit ein Umrechnungsfaktor, um diese in die Zieleinheit zu überführen.

Die absolute und relative Häufigkeit pro Fertigungslauf der Mengeneinheiten sind in der Tabelle 3.7 aufgeführt. Es fällt auf, dass anscheinend für dieselbe Einheit unterschiedliche

Tabelle 3.7: Mengeneinheiten und Häufigkeiten

Einheit	Anzahl Vorkommnisse	Durchschnittliche Häufigkeit pro Fertigungslauf
Stück	10327	81.9603
KG	8532	67.7143
	8527	67.6746
Liter	6505	51.6270
Gramm	6191	49.1349
Milliliter	2301	18.2619
ST	167	1.3254
Milligramm	121	0.9603
L	88	0.6984
µL	65	0.5159

Repräsentationen in der Datenbank vorliegen, wie es etwa bei Liter und L bzw. Stück und ST der Fall ist.

Eine Umrechnung von beispielsweise Litern in Kilogramm setzt Wissen über die Dichte der Flüssigkeit voraus. Die Dichte wiederum lässt sich mit Kenntnis von Substanz, Konzentration und Temperatur aus Tabellen entnehmen bzw. berechnen. Die Substanz ist theoretisch teilweise bekannt, kann jedoch aufgrund des Betriebsgeheimnisses nicht in dieser Arbeit verwendet werden. Über die Temperatur liegen keinerlei Informationen vor. Somit ist eine Umrechnung von Liter in Kilogramm und umgekehrt praktisch nicht möglich. Auch das Umwandeln von Stück in Kilogramm bzw. Liter setzt Wissen voraus, welches hier nicht vorliegt.

Um die unterschiedlichen Mengeneinheiten nun in eine zu überführen wird eine fiktive Einheit, im Folgenden als **unit** bezeichnet und mit der in Tabelle 3.8 dargestellten Umrechnungstabelle, definiert.

Über die gemeinsame Einheit **unit** ist es nun möglich, zu errechnen, wie viel eines Materials in den Fertigungsprozess einfließt, ohne dabei Fehler über unterschiedliche Einheiten wie Gramm und Kilogramm zu erhalten, da der Umrechnungsfaktor zwischen Gramm und Kilogramm durch die Umrechnung nach **unit** abgebildet wird. Selbiges gilt für alle anderen ineinander überführbaren Einheiten. Für das Umwandeln von Einheiten in nicht kompatible Einheiten, etwa Liter und Kilogramm, wurden willkürliche Schätzwerte angenommen. Ein besserer Ansatz für die Umrechnung von Liter nach Kilogramm wäre es, die durchschnittliche Dichte der verwendeten Substanzen zu verwenden.

Unter Verwendung dieses Ansatzes ergibt sich eine abweichende Repräsentation für Abbildung 3.8, welche in Tabelle 3.9 dargestellt ist.

Tabelle 3.8: Umrechnungsfaktoren für Mengeneinheiten

Einheit	Umrechnungsfaktor
Stück	1.0
KG	1.0
	1.0
Liter	1.0
Gramm	0.001
Milliliter	0.001
ST	1.0
Milligramm	0.000001
L	1.0
µL	0.000001

Tabelle 3.9: Ermittlungen aller einfließenden Mengen, **unit** basierter Ansatz

A	B	C	D	E	F	G	H	I	J
5.009	8.5	6.5	21	5	11	9	7.5	15	18

Im folgenden wird dieser Ansatz als „nicht naiver Ansatz“ bezeichnet.

Errechnung der konkreten, anteilig einfließenden Mengen

Bei dem vorherigen **unit** basierten Ansatz tritt jedoch das selbe Problem wie bei dem naiven Ansatz auf, es wird nicht immer die gesamte produzierte Menge eines Zwischenproduktes für die weitere Fertigung verwendet. Dies liegt daran, dass die produzierte Menge auch bei gleichen Ausgangsmengen ggf. auf Grund von nicht kontrollierbaren Faktoren leicht schwankt. Somit muss grundsätzlich mehr hergestellt werden als wirklich benötigt wird, um sicher zu stellen, dass genug für die Fertigung des Endproduktes bzw. des nächsten Teilproduktes verfügbar ist. Hierdurch entsteht ein Verlust von einfließendem Material.

Dieses Vorgehen ist erstmal unproblematisch, da theoretisch mittels einfachem Dreisatz ermittelt werden kann, wie viel der einzelnen Ausgangsmaterialien in einem Zwischenprodukt zu finden ist, und dieser Verlust somit ausgerechnet werden kann.

In der Praxis ist dies bei chemischen Prozessen jedoch nicht mehr möglich, da die Ausgangsstoffe nicht immer zu 100 Prozent in ein Zielprodukt umgewandelt werden, oftmals sind die Produkte chemischer Reaktionen Wasser oder Gase, wie etwa Kohlenstoffdioxid. Welche entsprechend nicht im gewünschten Zielprodukt vorhanden sind. Unter Kenntnis der genauen

Reaktionsgleichung ist es, durch Zuhilfenahme der molaren Masse (g/mol, das Gewicht pro $6.022 * 10^{23}$ Teilchen), möglich zu ermitteln, wie viel eines Zielproduktes bzw. wie viel Abfall entsteht.

Als Beispiel dient hier die Herstellung von Calciumsulfat (CaSO4) aus Calciumcarbonat (CaCO3) und Schwefelsäure (H2SO4), welche in Abbildung 3.9 dargestellt ist.

Abbildung 3.9: Reaktion von Calciumcarbonat und Schwefelsäure

$$1\,\mathrm{M}(100\,\mathrm{g})\mathrm{CaCO_3} + 1\,\mathrm{M}(98\,\mathrm{g})\mathrm{H_2SO_4} \longrightarrow 1\,\mathrm{M}(136\,\mathrm{g})\mathrm{CaSO_4} + 1\,\mathrm{M}(18\,\mathrm{g})\mathrm{H_2O} + 1\,\mathrm{M}(44\,\mathrm{g})\mathrm{CO_2}$$

In der Reaktionsgleichung aus Abbildung 3.9 wird deutlich, dass wider Erwarten nicht 100g oder 200g Feststoff, sondern 136g erstellt werden. Auch gehen 22% des Gesamtgewichtes verloren, da sie am Ende in Form von CO2 vorliegen. Hierdurch soll verdeutlicht werden, dass gerade bei chemischen Reaktionen, wie sie während des Prozesses auch stattfinden, nicht von einer Umwandlungseffektivität von 100%, basierend auf Gewicht oder Volumen, ausgegangen werden kann.

Auch müssen physikalische Prozesse, wie etwa Volumenkontraktion, bei der einfachen Mischung von Substanzen berücksichtigt werden, um präzise Ergebnisse zu liefern.

Eine genaue Ermittlung der eingeflossenen Bestandteile des Endproduktes ist somit nur mit unverhältnismäßig hohem Aufwand und einem umfangreichen domänenspezifischen Wissensschatz möglich.

Im Rahmen dieser Arbeit wird dieses Verfahren nicht weiter verfolgt. Eine Möglichkeit, einfacher vergleichbare Werte zu erreichen, ist es, über eine Umrechnungstabelle wie etwa Tabelle 3.8 alle Transaktionen auf eine einheitliche Mengeneinheit zu bringen und entsprechend über Dreisatz unter Annahme einer gewissen Effektivität einen Wert zu berechnen. Dieser Ansatz wird in Abschnitt 3.5.1 (Errechnung von geschätzt anteilig einfließenden Mengen) weiter verfolgt.

Errechnung von geschätzt anteilig einfließenden Mengen

Hier wird versucht, dass in Abschnitt 3.5.1 (Errechnung der konkreten, anteilig einfließenden Mengen) vorgestellte Verfahren für eine Effektivität von 100% zu realisieren.

Hierzu wird berechnet, wie viel von welchem Material in ein Zwischenprodukt einfließt. Über einfaches Aufsummieren wird ermittelt, wie viel von dem Zwischenprodukt hergestellt wird und anschließend wird ermittelt, wie viel dieses Produktes in die weitere Fertigung einfließt. Hierüber kann errechnet werden, wie viel Prozent des Zwischenproduktes verwendet wird,

Tabelle 3.10: Ermittlungen aller einfließenden Mengen, anteilig berechnet

Material	A	B	C	D	E	F	G	H	I	J	Produziert	Verwendet	Faktor
C	5	4	0	0	0	0	0	0	0	0	9	6.5	0.72
G	3.6	2.9	6.5	0	3.8	0	0	0	0	0	10.3	9	0.87
F	0	0	0	20	0.2	0	0	0	0	0	20.2	11	0.54
H	3.1	2.5	5.7	0	0.1	0	9	0	0	0	9	7.5	0.83
I	0.009	4.5	0	10,8	0.1	11	0	0	0	0	15.509	15	0.96
J	2.6	6.4	4.7	10.3	0.2	10.6	0.8	7.5	15	0	23.5	18	0.77
Prod	2.0	7.9	3.6	8.9	0.2	8.2	0.6	5.8	11.6	18	18		

und anschließend können dann die Mengen der eingehenden Materialien mit dem Prozentwert multipliziert werden, um zu ermitteln, wie viel des Materials nun konkret einfließt. Dieses Verfahren wird beispielhaft in Tabelle 3.10 dargestellt.

Bei ersten Experimenten mit den Realdaten ist an dieser Stelle aufgefallen, dass nicht alle Materialien erfasst werden, welche in ein Produkt oder Zwischenprodukt einfließen. Daher wird teilweise eine Effektivität von > 1.0 (100%) errechnet, zum Beispiel wird bei dem Ansetzen einer Lösung darauf verzichtet, die Menge an einfließendem Wasser zu dokumentieren. Dadurch kann nicht nachvollzogen werden, mit welchen Anteilen die Bestandteile der Lösung weiter verwendet werden. Aufgrund dieser Problematik ist es nicht möglich, die Anteile eines Zwischenproduktes an dem Endprodukt auch nur näherungsweise zu berechnen.

3.5.2 Abbildung auf Sequenzen

Eine weitere Möglichkeit den Chargenbaum in ein geeignetes Format zu überführen, ist die Abbildung auf einer Sequenz. Zwar können Verfahren, wie MLP oder Entscheidungsbäume, nicht mit Sequenzen unterschiedlicher Länge umgehen, doch sind LSTMs für dieses Format geeignet. LSTMs sind zwar theoretisch gut für die Klassifikation von Sequenzen geeignet, jedoch benötigen sie in der Praxis meist recht große Datenmengen, welche hier leider nicht vorliegen. Dennoch werden im Rahmen dieser Arbeit LSTMs betrachtet, um ihren potentiellen Nutzen zu erproben.

Auch hier gibt es verschiedenste Ansätze. Im Folgenden wird, wie auch bei den in Abschnitt 3.5.1 beschriebenen Ansätzen, auf eine Verwendung der Aktivitäten verzichtet. Für die Kodierung werden zwei grundsätzlich unterschiedliche Ansätze betrachtet. Einerseits wird das Endprodukt und andererseits werden die Blätter als Startpunkt gewählt. Für den Aufbau der Sequenz wird jedoch in beiden Fällen jeweils Breiten- und auch Tiefensuche verwendet, um unterschiedliche Reihenfolgen zu vergleichen.

Die Wurzel als Startpunkt

Die Sequenz startet bei der Wurzel des Baumes. Dies hat den Vorteil, dass es recht einfach ist, Sequenz aufzubauen, da der Startpunkt bekannt ist. Grundsätzliche können verschiedene Verfahren zum Aufbau der Sequenz verwendet werden, an dieser Stelle werden sowohl Breiten- als auch Tiefensuche anhand des Baumes aus Abbildung 3.8 betrachtet.

Gibt es jedoch eine Abzweigung im Baum, so muss entschieden werden, welcher der möglichen Pfade an führender Stelle in der Sequenz auftaucht. An dieser Stelle werden zwei Möglichkeiten betrachtet, einerseits wird der kürzeste Pfad und andererseits der längste Pfad an erster Stelle in der Sequenz aufgeführt.

Es gibt jedoch ein Problem mit diesem Ansatz. Um die Wurzel des Baumes, also das Endprodukt der Fertigung, als Startpunkt der Sequenz verwenden zu können, muss der Fertigungsprozess abschlossen sein. Hierdurch wird verhindert, dass Vorhersagen für nicht abgeschlossene Prozesse getätigt werden können.

Breitensuche Wird der Baum aus Abbildung 3.8 unter Verwendung von Breitensuche mit dem kürzestem Pfad als erstes in der Sequenz zu einer Sequenz umgewandelt, ergibt sich das Ergebnis, welches in Listing 3.8 dargestellt ist.

Listing 3.8: Sequenz nach Breitensuche mit kürzestem Teilbaum zuerst

```
1 (D,1),(J,18),(E,1),(I,15),(H,7.5),(B,4.5),(A,0.009),(F,11),
2 (G,9),(D,20),(E,0.2),(E,3.8),(C,6.5),(A,5),(B,4)
```

Eine Alternative hierzu wäre es, als erstes den längsten Teilbaum bei einer Verzweigung zu wählen, hiermit ergäbe sich die Sequenz aus Listing 3.9.

Listing 3.9: Sequenz nach Breitensuche mit längstem Teilbaum zuerst

```
1 (J,18),(D,1),(H,7.5),(I,15),(E,1),(G,9),(F,11),(B,4.5),
2 (A,0.009),(C,6.5),(E,3.8),(D,20),(E,0.2),(A,5),(B,4)
```

Tiefensuche Im Gegensatz zur Breitensuche wird in der Tiefensuche ein Pfad im Baum jeweils vollständig bis zu allen seinen Blättern abgebildet. Welches Verfahren besser für das Arbeiten mit einem LSTM geeignet ist, wird im Kapitel Datamining (Abschnitt 3.6) geklärt.

Auch hier gibt es wieder die zwei unterschiedlichen Ansätze den längeren oder den kürzeren Teilbaum an erster Stelle der Sequenz zu haben.

Wird der längere Teilbaum als erstes dargestellt, ergibt sich die Sequenz, welche in Listing 3.10 abgebildet ist.

Listing 3.10: Sequenz nach Tiefensuche mit längstem Teilbaum zuerst

```
(J,18),(H,7.5),(G,9),(C,6.5),(A,5),(B,4),(E,3.8),(I,15),
(F,11),(D,20),(E,0.2),(B,4.5),(A,0.009),(E,1),(D,1)
```

Wird statt dem längstem der kürzeste Teilbaum als erstes in der Sequenz eingebunden, ergibt sich eine abweichende Sequenz, siehe Listing 3.11. Im Rahmen des Dataminings wird geprüft, ob diese unterschiedliche Kodierung abweichende Ergebnisse liefert.

Listing 3.11: Sequenz nach Tiefensuche mit kürzerem Teilbaum zuerst

```
(D,1),(J,18),(E,1),(I,15),(B,4.5),(A,0.009),(F,11),(D,20),
(E,0.2),(H,7.5),(G,9),(E,3.8),(C,6.5),(A,5),(B,4)
```

Die Blätter als Startpunkt

Das Problem, dass der Prozess abgeschlossen sein muss, um eine Analyse durchzuführen, wird durch den Ansatz, die Blätter als Startpunkt der Sequenz zu verwenden, beseitigt. Jedoch ergibt sich hierdurch das Problem, dass alle Blätter bekannt sein müssen. Alle relevanten Blätter lassen sich beispielsweise aus den Zwischenprodukten ermitteln. Ist etwa die Fertigung aus Abbildung 3.8 bis vor das Zwischenprodukt J abgeschlossen, wie in Abbildung 3.10, so können alle Blätter über H, E und I mittels Baumtraversion ermittelt werden. Hierfür muss allerdings bekannt sein, welches konkrete Zwischenprodukt in der weiteren Fertigung verwendet werden soll. Dies ist allerdings, insbesondere wenn mehrere Prozesse in einer Fertigungsanstalt parallel betrieben werden, nicht immer ohne weiteres gegeben.

Jedoch stellt sich hier erst recht die Frage, in welcher Reihenfolge die Knoten und Blätter in die Sequenz geschrieben werden sollen.

Da jedoch bei den hier vorliegenden Daten jeweils die vollständig abgearbeiteten Chargenbäume vorliegen, wird an dieser Stelle auf ein direktes Aufbauen von den Blättern aus verzichtet. Stattdessen werden die Sequenzen, welche auf Basis der Wurzel als Startpunkt aufgebaut wurden, lediglich in ihrer Reihenfolge umgekehrt, um einen Start auf Basis der Blätter zu simulieren. Sollte diese Art von Sequenzen im Datamining Schritt erfolgversprechende Ergebnisse liefern, werden Teile des Endes eben dieser Sequenzen entfernt, um das Arbeiten auf einem nicht abgeschlossenen Chargenbaum zu simulieren.

3.5.3 Kodierung von nicht numerischen Werten

Im Falle der Ermittlung von einfließenden Teilmengen, wie in Abschnitt 3.5.1 beschrieben, sind lediglich numerische Werte für die Mengen der einfließenden Materialien vorhanden. Es

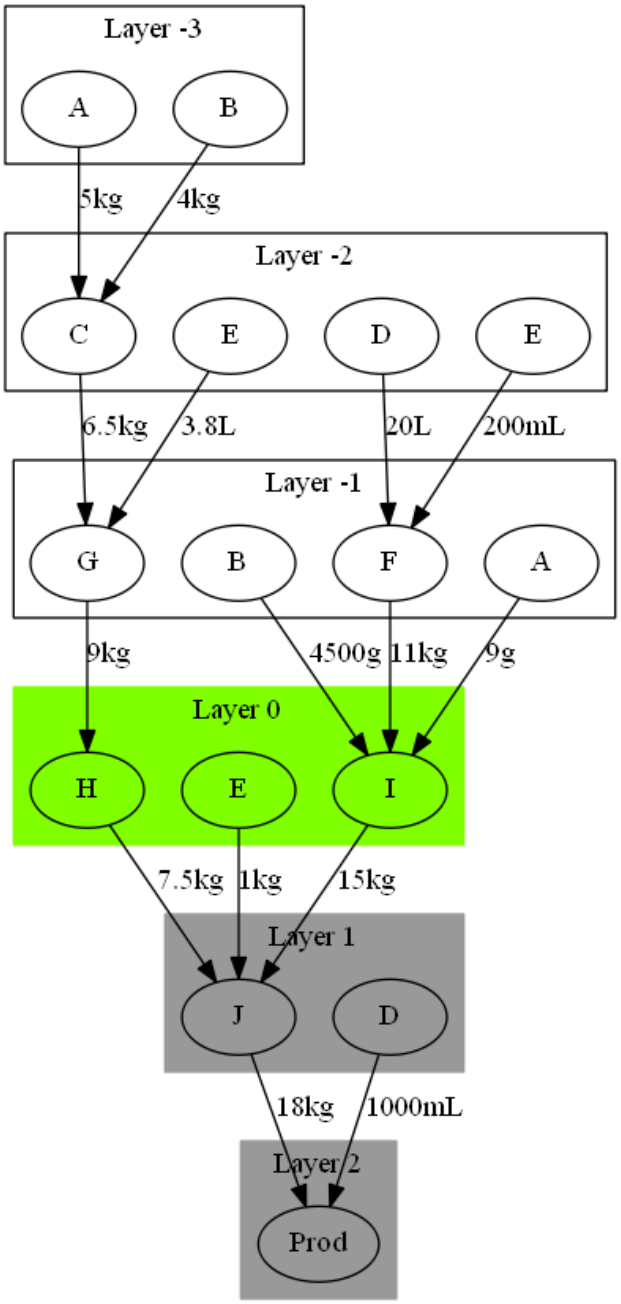

Abbildung 3.10: Einfließende Teilmengen Demo tree, partiell abgearbeitet

muss also keine besondere Art der Darstellung dieser verwendet werden. Bei der Abbildung auf Sequenzen muss jedoch auch neben der Menge das Material kodiert werden, welches einfließt. Hierfür bieten sich verschiedene Formen an. Zwei gängige Verfahren zur Kodierung von nominalen Werten werden im Folgendem kurz vorgestellt. Eine Möglichkeit der Kodierung der Materialien ist etwa eine Abbildung auf numerische Werte. Bei den ursprünglichen Daten handelt es sich um nominale Daten ohne jegliche Ordnung, durch eine Abbildung auf numerische Werte, beispielsweise die natürlichen Zahlen, wird jedoch mindestens eine Ordnungsrelation impliziert, was die Bedeutung der Daten verfälscht. Somit ist diese Art der Kodierung weniger gut geeignet.

Eine Alternative zu diesem Verfahren wäre etwa die One-hot Kodierung, welche sich insbesondere zur Darstellung von nominalen Werten eignet. Das Attribut wird hierbei auf einen N-dimensionalen Vektor abgebildet, wobei N der Anzahl der verschiedenen Ausprägungen des Attributes entspricht. Dieser Vektor enthält die Werte 0 und 1, die Summe aller Elemente entspricht hierbei immer 1.

Die One-Hot Kodierung ist geeignet zur Darstellung von nominalen Werten, da keine Ordnungsrelation oder ähnliches impliziert wird, jedoch wird ein zu kodierendes Attribut auf eine teilweise sehr hohe Anzahl von Attributen abgebildet. Zu beachten ist auch, dass diese Form der Kodierung nur bei endlichem Wertebereich, welcher beim Zeitraum der Entwicklung des Modells vollständig bekannt sein muss, möglich ist. Sollte bei Verwendung eines Modells Datenattribut-Ausprägungen auftreten, welche zum Zeitpunkt der Erstellung nicht berücksichtigt wurden, so ist es nicht möglich, diese Daten mittels des Modells zu bearbeiten.

Bei Materialien, welche in Fertigungsprozessen verwendet werden, besteht jedoch oft eine Austauschbarkeit der Materialien, sodass ein Material in der Fertigung beispielsweise durch ein Anderes substituiert oder ergänzt werden kann. Dieses Konzept kann über die Verwendung einer One Hot Kodierung nicht ohne weiteres eingebracht werden, und bedarf ggf. einer andere Kodierung.

Aufgrund der Nachteile des ersten Verfahrens wird für die praktischen Experimente mit LSTMs eine One-Hot Kodierung für die Materialien verwendet.

3.5.4 Abbildung von Aktivitäten

Auf die Verwendung von Aktivitäten wird im Mining Prozess in dieser Arbeit verzichtet. Dies hat mehrerer Gründe, es war nicht möglich diese ausreichend zu anonymisieren und außerdem ist die vorhandene Datenbasis bezüglich der Aktivitäten stark unvollständig, was auf die defekte CSV Datei, in welcher Aktivitäten geliefert werden sollten, zurückzuführen ist. Auch gibt es eine Vielzahl von stark verschiedenen Aktivitäten, dessen Auswertung jeweils exakt

auf die Art von Aktivität angepasst werden muss. Dieser Aufwand würde den Rahmen dieser Arbeit sprengen. Des Weiteren beschäftigt sich ein Kollege im Rahmen seiner Masterarbeit mit der Auswertung einiger ausgewählten Aktivitäten.

3.5.5 Datenstandardisierung und -normalisierung

Wie bereits in Abschnitt 2.1.3 erwähnt, gibt es verschiedene Techniken, welche geeignet sind, um numerische Werte eines beliebigen Wertebereiches auf den Bereich von 0...1 bzw. -1...1 abzubilden. In dieser Arbeit werden die zwei Verfahren -Standardisierung und Normalisierung- verwendet und die Ergebnisse der Modelle, welche mit den entsprechend transformierten Daten trainiert wurden, verglichen.

Stand des Prozesses

Nach dem Schritt der Transformation liegen jetzt verschieden transformierte vor verarbeitete Daten vor. Diese liegen in Form von CSV Dateien vor und sind geeignet, um verschiedene ML Verfahren auf diese Artefakte des Prozesses anzuwenden. Abbildung 3.11 zeigt die Stelle, an welcher diese Daten in dem KDD Prozess vorliegen.

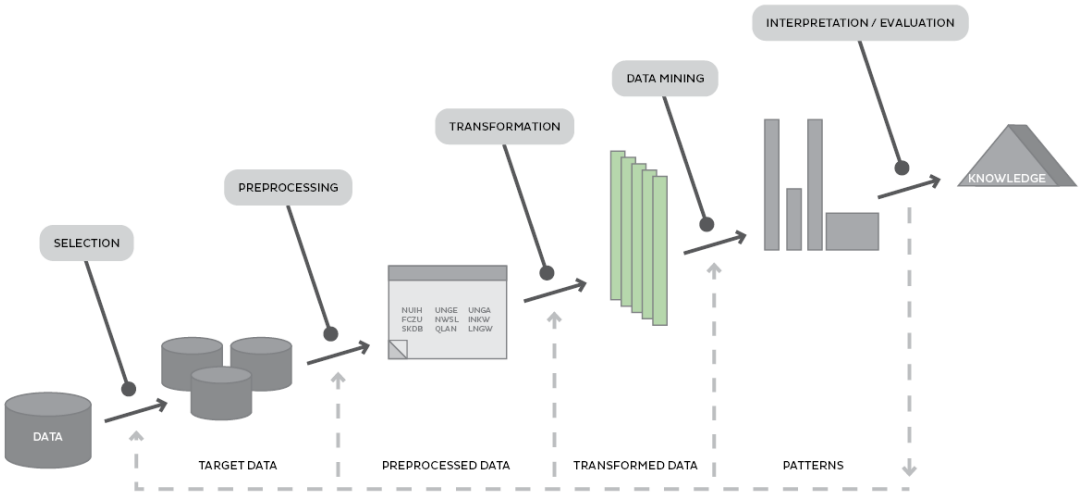

Abbildung 3.11: KDD Prozess - Transformed Data

3.6 Data Mining

Alle hier betrachteten Verfahren werden an dieser Stelle als Supervised Learning Methoden verwendet. Um ein Overfitting des Modells zu verhindern bzw. zu ermitteln, wann ein solches eintritt, erfolgt eine Aufteilung der Daten nach der soweitigen Aufbereitung in trainings und test Data Set. Hierbei werden verschiedene Aufteilungen erprobt. Insgesamt werden, bei jedem der Data Mining-Verfahren, neun verschiedene Verhältnisse zwischen Trainings- und Test-Daten erprobt: 10/90, 20/80, 30/70, 40/60, 50/50, 60/40, 70/30, 80/20 und 90/10.

Bei der Aufteilung wird ein statistisches Samplen durchgeführt, sodass jede Klasse in beiden Sets gleich stark vertreten ist. Die Aufteilung erfolgt einmalig, sodass alle Verfahren unter den selben Bedingungen erprobt werden können, und die Auswirkungen von unterschiedlichen Aufteilungen von Trainings- sowie Test-Data auf die Ergebnisse beseitigt werden. Für die Bewertung der Mining-Ergebnisse wird eine einfache Fehlerrate sowie das Konzept der Confusion Matrix herangezogen und grafisch dargestellt.

Zu erwähnen ist noch, dass für alle Verfahren, welche auf neuronalen Netzen basieren, der in Tensorflow implementierte Adams Optimizer verwendet wird. Der Algorithmus ist in [KB14] beschrieben, und wird mit den dortigen Empfehlungen angewandt.

3.6.1 Entscheidungsbäume mittels C4.5

Unter Verwendung des C4.5 Algorithmus werden sämtlich unter Abschnitt 3.5.1 beschriebenen und für realisierbar befundenen Verfahren betrachtet. Die numerischen Mengen an einfließenden Materialien werden einerseits nicht vor verarbeitet, andererseits jedoch auch unter Verwendung von Binning mit verschieden vielen Binns kodiert. Bezüglich des Binnings wurden zwei unterschiedliche Anzahlen von Binns verwendet. Es wurden drei Binns mit jeweils gleicher Breite und fünf Binns mit gleicher Breite verwendet.

Für sämtliche Experimente, welche den C4.5 Algorithmus umfassen, wird die im Rahmen von Knime bereitgestellte Implementation des Algorithmus verwendet. Von einer Eigenimplementierung wird hier abgesehen, um Implementationsfehler zu vermeiden. Auch ist dies schlicht nicht notwendig, da bereits viele erprobte Implementationen online verfügbar sind.

Für eine kurze übersichtliche Darstellung der Ergebnisse wird an dieser Stelle die Fehlerrate als Metrik verwendet, und in Abbildung 3.12 dargestellt. Eine konkretere Auswertung auf der Fehlerrate und des Recalls der beiden Klassen erfolgt im späteren Abschnitt 3.7. Die Abbildung 3.12 hat nicht den Anspruch eine präzise Bewertung des Verfahrens auf den hier verwendeten Daten zu liefern. Sie liefert lediglich eine Übersicht über die potentielle Leistungsfähigkeit des C4.5 Algorithmus auf diesen Daten. Der C4.5 Algorithmus ist in in der Lage, auf

allen Arten von hier verwendeten Daten ab einer Aufteilung von Training- zu Testdaten von 60% zu 40% einen Fehler von unter 20% zu erreichen.

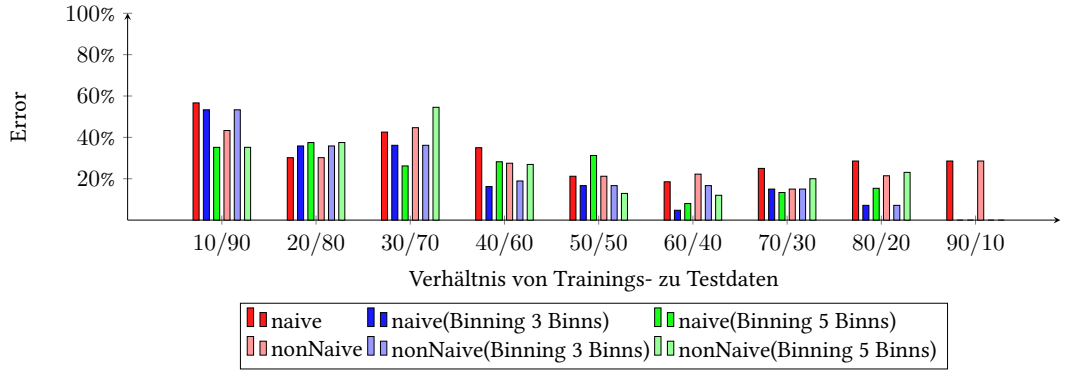

Abbildung 3.12: C 4.5, Naive vs. nonNaive

3.6.2 Perzeptron

Wie auch beim C4.5 Algorithmus werden an dieser Stelle sämtliche realisierbare Verfahren aus Abschnitt 3.5.1 unter verschiedenen Arten der Codierung der Mengen verwendet. Konkret werden die Ansätze der Standardisierung und Normalisierung gegenübergestellt. Als Aktivierungsfunktion wird die Softmax Funktion verwendet.

Bei ersten Experimenten wurde festgestellt, dass die Netze nicht immer konvergieren. Um diesem Problem aus dem Weg zu gehen, wurden von jeder Architektur 1000 Netze mit unterschiedlich initialisierten Bios und Waight Werten erstellt und jeweils für 1500 Epochen trainiert. In der Abbildung 3.13 ist der durchschnittliche sowie maximale als auch minimale Fehler dargestellt, den die Perzeptron Netze für die entsprechenden Aufteilungen der Daten generieren. Diese Abbildung zeigt, dass ein Perzeptron bereits in der Lage ist, relativ geringe Klassifikationsfehler zu erzielen. Die Daten sind somit bis zu einem gewissen Grad linear trennbar.

3.6.3 Multi Layer Perzeptron

Hier gelten dieselben Rahmenbedingungen wie bei der Arbeit mit einem einfachen Perzeptron wie in Abschnitt 3.6.2 beschrieben. Ergänzend kommt hinzu, dass Netze mit unterschiedlichen Anzahlen von Hidden Layern sowie unterschiedlichen Ausdehnungen, also Breiten, eben dieser Schichten erprobt werden. Auf Ebene der Hidden Layer wird die Sigmoid Funktion als

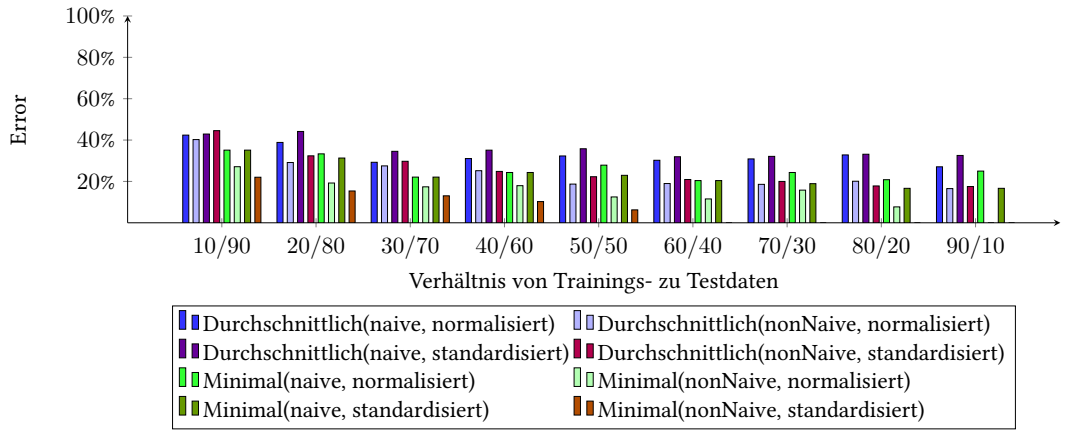

Abbildung 3.13: Perzeptron Naive vs. nonNaive, normalisiert vs. standardisiert (1500 Trainingsepochen)

Aktivierungsfunktion verwendet. An der Ausgabeschicht wird wie auch bei dem einfachen Perzeptron die Softmax Aktivierungsfunktion verwendet.

Die zwei Diagramme in Abbildung 3.15 und Abbildung 3.15 zeigen, dass ein MLP potentiell bessere Ergebnisse liefert als ein einfaches Perzeptron. Jedoch ist auch zu beobachten, dass ein komplexeres Modell nicht zwangsläufig bessere Ergebnisse liefert.

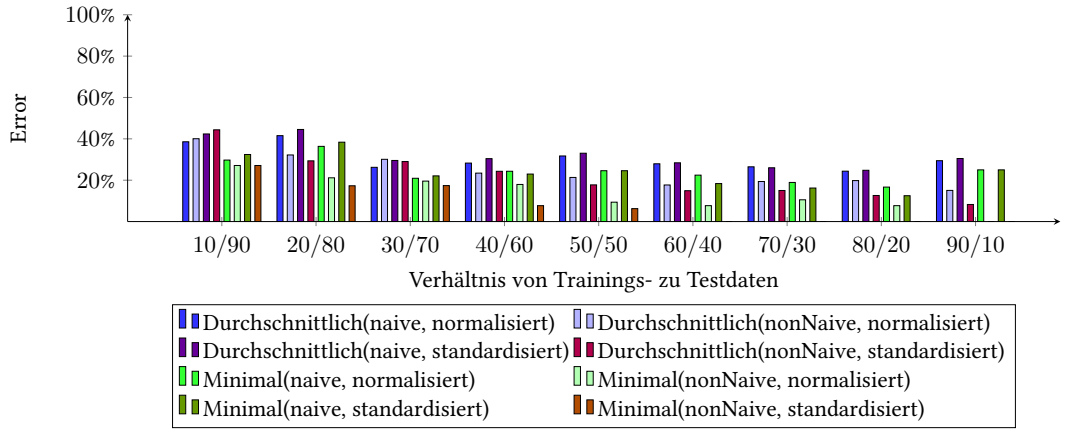

Abbildung 3.14: Multi Layer Perzeptron Naive vs. nonNaive (eine Hidden Layer mit einer Breite von 100, 1500 Trainingsepochen)

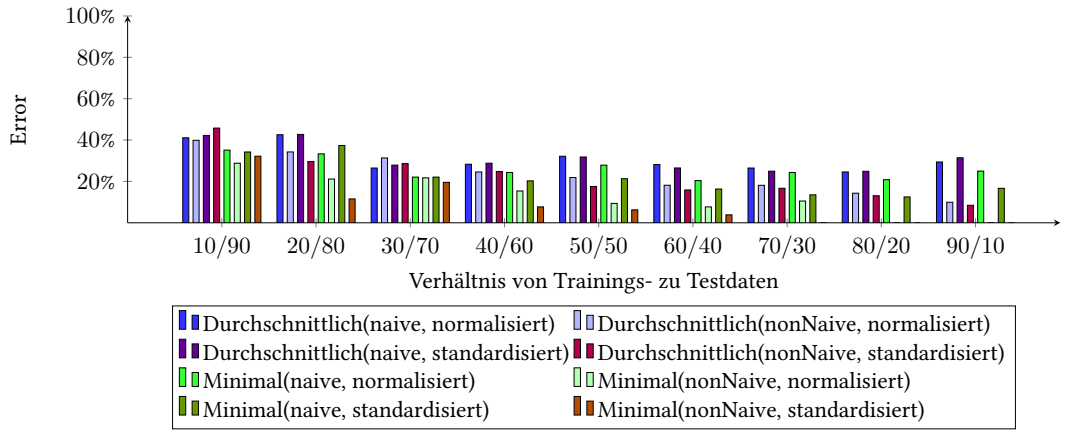

Abbildung 3.15: Multi Layer Perzeptron Naive vs. nonNaive (zwei Hidden Layer mit einer Breite von je 100, 1500 Trainingsepochen)

3.6.4 Long Short Term Memory Network

Die Struktur der Daten, welche für das Trainieren des LSTMs verwendet werden, weicht stark von den anderen Ansätzen ab. Dies ist darauf zurückzuführen, dass LSTMs bevorzugt auf Sequenzen von Daten arbeiten, daher die Chargenbäume, wie in Abschnitt 3.5.2 beschrieben, auf Sequenzen abgebildet werden. Jedoch müssen wesentlich mehr Materialien kodiert werden, da auch Materialien, welche bei der bisherigen Analyse aufgrund der Menge von Null nicht betrachtet wurden, hier zum Tragen kommen.

Innerhalb des LSTM Netzes kommen fünf LSTM Zellen zum Einsatz. Auch in diesem Falle konnte keine Konvergenz des Modelles beobachtet werden, sodass wieder sämtliche Messungen über eine Anzahl von Modellen ermittelt und verrechnet wurden.

Die Abbildung 3.16 und die Abbildung 3.17 zeigen beide, dass ein LSMT unter den hier gegebenen Bedingungen bei steigender Anzahl von Trainingsdaten eher schlechtere Ergebnisse liefert.

Stand des Prozesses

Durch Anwendung von ML Verfahren wurden die bereits transformierten Daten in Modelle bzw. Muster umgewandelt. Der Abbildung 3.18 lässt sich ihre Position entnehmen. Die Modelle bzw. Muster sind die vorletzte Art von Artefakten, welche durch den KDD Prozess generiert werden. Im Folgenden muss aus ihnen durch eine Interpretation lediglich Wissen generiert werden, was mit einigen Herausforderungen verbunden ist.

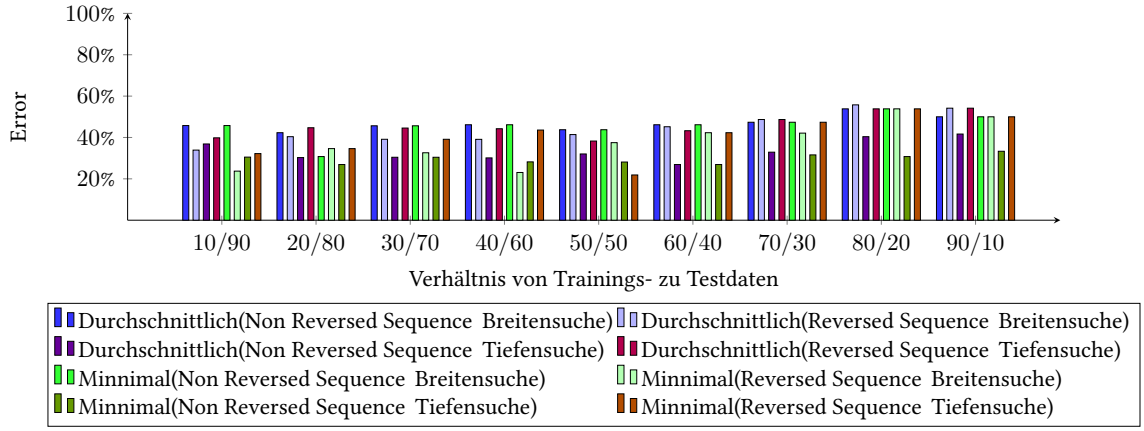

Abbildung 3.16: LSTM, unter Verwendung von Normalisierung (10000 Trainingsepochen, Batch Size=1)

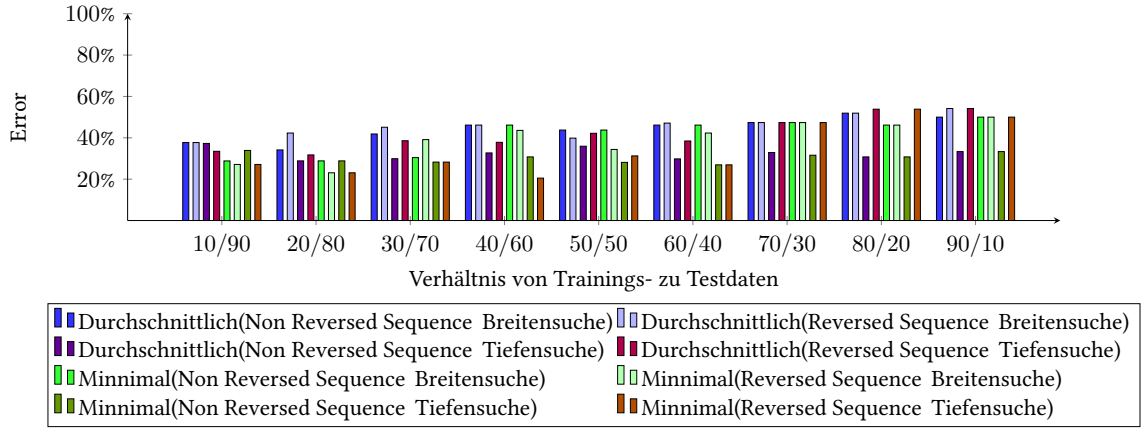

Abbildung 3.17: LSTM, unter Verwendung von Standardisierung (10000 Trainingsepochen, Batch Size=1)

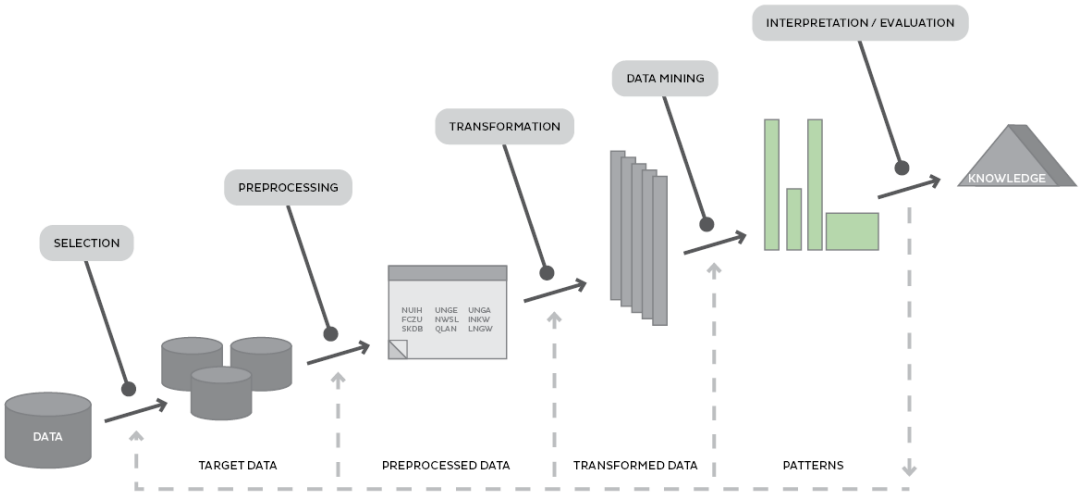

Abbildung 3.18: KDD-Prozess - Patterns

3.7 Interpretation

Dieses Kapitel befasst sich mit der Auswertung der Ergebnisse des Data Mining Schrittes. Hierbei wird auch auf die Sinnhaftigkeit und den Nutzen der Modelle eingegangen. Es folgt eine Gegenüberstellung der einzelnen Verfahren anhand von objektiven Metriken, und es wird über den möglichen Nutzen der Ergebnisse spekuliert. Aufgrund der geringen Menge an Daten sind jedoch die hier ermittelten Zahlen nicht zwangsläufig auf größere Datenmengen generalisierbar.

Die Konkreten Ergebnisse der Experimente finden sich Tabellarisch dargestellt in dem Kapitel 5, sie ergänzen die hier durchgeführte Interpretation.

3.7.1 Gegenüberstellung bezüglich Klassifikationsgenauigkeit

Im Allgemeinen lässt sich beobachten, dass alle getesteten Vertreter von Data Mining Verfahren mehr oder weniger erfolgreich waren. Dieser Abschnitt bezieht sich auf die Ergebnisse, welche in den Tabellen in Abschnitt 5 dargestellt sind.

C4.5

Der C4.5 Algorithmus ist in der Lage, auf den hier vorliegenden Daten Fehlerraten von unter 20% zu erreichen. Dies trat bei jeder Art von Vorverarbeitung bei einem Anteil der Trainingsdaten von 60% ein.

Die besten Ergebnisse werden durch den nicht naiven Ansatz unter Verwendung von Bucketing, sowie den naiven Ansatz mit fünf Buckets bei jeweils 90% Trainingsdaten erreicht. Beide Ansätze erreichen eine Fehlerrate von 0%. Dieses Ergebnis ist aufgrund der geringen Menge an Testdaten nicht repräsentativ.

Die zweitbesten Ergebnisse liefert der naive Ansatz bei 60% Trainingsdaten unter Verwendung von Bucketing mit drei Buckets. Die erreichte Fehlerrate liegt in dem Fall bei 4.76%. Im Algemeinen kann beobachtet werden, das die Verwendung von Binning bis zu einem gewissen Punkt den Fehler eines Modells Reduzieren kann, bei zu vielen Binns steigt der Fehler jedoch wieder.

Weder der Recall Wert der unterdurchschnittlichen Klasse noch der der überdurchschnittlichen Klasse sind unausgewogen stark ausgeprägt, sodass Fehlklassifikationen nicht in eine Richtung tendieren.

Perzeptron

Unter Betrachtung aller Ergebnisse des Perzeptrons fällt auf, dass die besten Ergebnisse unter Verwendung des nicht naiven Ansatzes ab einem Trainings- zu Testdaten Verhältnis von 60/40 erzielt werden. Jedoch fällt auch auf, dass der durchschnittliche Fehler bei normalisierten Daten geringer ist als bei standardisierten Daten. Dennoch liefern Modelle, welche unter Verwendung von Standardisierung entwickelt wurden, potenziell bessere Ergebnisse. Des Weiteren ist in allen vier Experimenten der durchschnittliche bzw. maximale Recall der unterdurchschnittlichen Klasse höher und somit besser als der entsprechende Recall der überdurchschnittlichen Klasse. Das bedeutet, dass eine Tendenz zur Fehlklassifikation als unterdurchschnittlich vorliegt.

Das beste Ergebnis mit einem Error von 0% wird unter Verwendung von Standardisierung und dem nicht naiven Ansatz ab einer Aufteilung von 60% zu 40% erzielt. Bei weiterer Reduzierung der Testmenge fällt der durchschnittliche Fehler weiter ab.

Multi Layer Perzeptron

Ein Multi Layer Perzeptron, mit einer Hidden Layer mit der Ausdehnung 100, erreicht im direktem Vergleich mit einem Perzeptron, abgesehen von dem naiven Ansatz mit Standardisierung, bessere Ergebnisse. Wieder besteht eine Tendenz zu einem höherem Recall der unterdurchschnittlichen Klasse.

Wieder liefert das Modell, welches unter vewendung des nicht naiven Ansatzes und Standardisierung trainiert wurde, die besten Ergebnisse mit einem Error von 0% ab einer Aufteilung von 60/40.

Ein neuronales Netz mit zwei Hidden Layern und je 100 Neuronen pro Hidden Layer verhält sich ähnlich, jedoch ist die Ungleichheit der zwei durchschnittlichen Recall Werte stärker ausgeprägt. Auch werden tendenziell bei mehr Trainingsdaten ab einer Aufteilung von 70/30 gute Error und Recall Werte erreicht. Wieder werden die besten Ergebnisse, ein Fehler von 0%, mit Standardisierung und dem nicht naiven Ansatz erreicht.

LSTM

Die Ergebnisse der LSTM Netze zeigen einen starken Unterschied zwischen umgekehrten und nicht umgekehrten Sequenzen. Die nicht umgekehrten Sequenzen weisen eine starke Tendenz zu einem wesentlich höherem Recall der unterdurchschnittlichen Klasse auf, verglichen mit dem der überdurchschnittlichen Klasse.

Die besten Ergebnisse der LSTM Experimente stammen von einem Netz, welches unter Verwendung einer umgekehrten, mit Tiefensuche erstellten Sequenz, unter Verwendung von Normalisierung trainiert wurde. Es erreichte bei 50% Testdaten einen Fehler von 21.875% bei einem Überdurschnitts-Recall von 0.714 sowie einem Unterdurchschnitts-Recall von 0.833.

Im Allgemeinen sind die Ergebnisse der LSTMs nicht in der Lage, den hohen Berechnungsaufwand zu rechtfertigen, was jedoch auch an der geringen Datenmenge liegen kann.

Modell übergreifend

Für einen Modellübergreifenden Vergleich werden lediglich Trainings- zu Testdaten Verhältnisse von 70/30 und weniger Trainingsdaten verglichen. Dies wird gemacht, da lediglich 68 Chargenbäume verfügbar sind. Da die Ergebnisse möglichst repräsentativ sein sollten, wurde als unteres Limit der Größe der Testdatenmenge 30% angenommen. Dies entspricht 23 Datensätzen.

Unter den zu betrachtenden Experimenten wurde der geringste Fehler von 0% von zwei Verfahren unter gleich vor verarbeiteten Daten erzielt. Sowohl ein Multi Layer Perzeptron mit einer Hidden Layer von 100 Neuronen als auch ein einfaches Perzeptron erreichten ab 60% Trainingsdaten unter Verwendung von Standardisierung und dem nicht naiven Ansatz einen Fehler von 0%. Der durchschnittliche Fehler liegt bei dem Perzeptron bei 20.938%, das MLP erreicht einen durchschnittlichen Fehler von 14.923% und ist somit tendenziell das bessere Modell. Beide Modelle können somit bessere Ergebnisse liefern als der C4.5 Algorithmus, welcher im besten Fall einen Fehler von 4.76% lieferte.

Die Modelle, welche auf einer LSTM Architektur basieren, liefern durchweg schlechtere Ergebnisse als die anderen Modelle.

Zusammenfassung

Im Zuge der Experimente und der Auswertung dieser wurde festgestellt, dass der nicht naive Ansatz der Daten Transformation zu bevorzugen ist, auch konnte ein deutlicher Vorteil durch Verwendung von Standardisierung statt Normalisierung festgestellt werden.

Die erfolgversprechendsten Modelle zur Auswertung dieser konkreten Daten sind einfache neuronale Netze. Wohingegen die Verwendung eines LSTM Netzes zumindest unter den hier geltenden Bedingungen nicht lohnend ist. Sollte doch ein LSTM Netz verwendet werden, so liefert zumindest in diesen Experimenten, eine Sequenz welche von den Blättern des Chargenbaumes aufgebaut wurde bessere Ergebnisse.

3.7.2 Objektive Metriken

In dem Abschnitt 2.1.5 dieser Arbeit wurden objektive Metriken als Teil eines Bewertungsschrittes des KDD Prozesses vorgestellt. Jedoch ist es bis dato zu keiner Rücksprache mit dem Kunden gekommen, sodass diese Metriken nicht akkurat angewendet werden können. Aus diesem Grunde kann nur über den Nutzen der Modelle spekuliert werden.

3.7.3 Nutzen

Es ist zwar leider nicht zu einer Rücksprache mit dem Kunden gekommen, dennoch kann an dieser Stelle über einen potenziellen Nutzen der Ergebnisse dieser Arbeit geschrieben werden.

Es scheint so, als seien alleine in den reinen Chargenbaumdaten Merkmale bzw. Kombinationen von Merkmalen vertreten, welche einen überdurchschnittlichen von einem unterdurchschnittlichem Fertigungslauf unterscheiden. Daraus lässt sich schließen, dass anscheinend die Mengen der einfließenden Materialien die Qualität eines Endproduktes maßgeblich beeinflussen. Somit wurde gezeigt, dass der zugrunde liegende Fertigungsprozess optimiert werden kann, um die Schwankungen der Qualität zu reduzieren und diese nicht ausschließlich durch nicht erfasste oder erfassbare Werte beeinträchtigt wird.

Hierdurch ist ein sukzessiver Optimierungsprozess durch das Unternehmen möglich. Es können auch über wiederholten Aufbau eines Modells zu unterschiedlichen Schritten in diesem Optimierungsprozess das Potential von Anpassungen der in das Modell einfließenden Parameter ermittelt werden. Hat das Modell einen Fehler von 50%, so kann davon ausgegangen werden, dass nicht wirklich Optimierungspotential auf Basis der verwendeten Attribute möglich ist [CL16].

Der C4.5 Algorithmus, welcher für die Entscheidungsbäume genutzt wurde, lässt direkte Rückschlüsse auf die Drehschrauben zur Optimierung des Fertigungsprozesses zu. Zusätzlich

könnte versucht werden, aus den anderen Modellen, über Auswertung der internen Gewichtungen, Informationen über relevante Materialien zu gewinnen. Dies ist bei der Perzeptron Architektur ein realistisches Vorgehen, jedoch ist unklar inwieweit mit diesen Informationen gearbeitet werden kann. Bei komplizierteren Netzen ist eine solche Auswertung nicht wirklich gut möglich.

Stand des Prozesses

Der KDD Prozess ist nun vollständig durchlaufen, und es liegt Wissen in Form von Modellen vor, welche für eine Optimierung der Fertigung genutzt werden könnten.

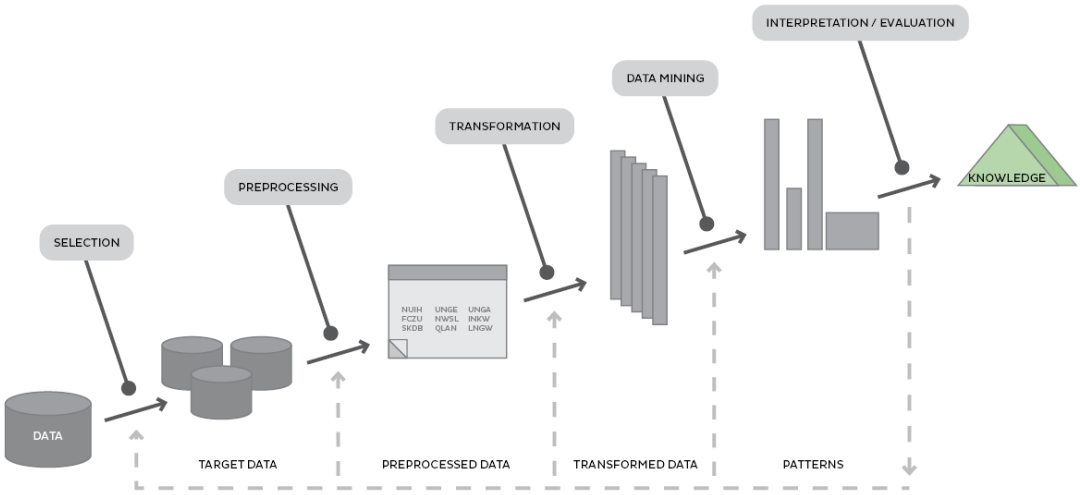

Abbildung 3.19: KDD Prozess - Knowledge

4 Fazit

4.1 Zusammenfassung

Ziel dieser Arbeit ist die Betrachtung des KDD Prozesses zur Verarbeitung von Complex Objects sowie die Erprobung einfacher Data Mining Verfahren, um lohnenswerte Ansätze für eine umfassendere Analyse der Daten zu offenbaren. Es stellt sich schnell heraus, dass der KDD Prozess zwar gut geeignet, die Qualität der bereitgestellten Daten aber nicht optimal ist. Insbesondere kurze oder leere Chargenbäume würden sich durch eine bessere Datenselektion wahrscheinlich vermeiden lassen, um die Anzahl der zur Verfügung stehenden Daten zu erhöhen.

Auch weisen die Aktivitätsdaten eine hohe semantische Komplexität auf, weshalb sie in dieser Arbeit nicht weiter betrachtet werden. Insbesondere Freitextfelder innerhalb der Aktivitäten erschweren neben der hohen Anzahl verschiedener Typen die Auswertung dieser, da ein Arbeiten mit beliebigen Textwerten einiges an zusätzlichem Aufwand mit sich bringt.

Der gesamte KDD Prozess wurde erfolgreich durchlaufen, und auch vielversprechende erste Ergebnisse mit einfachen Data Mining Ansätzen erreicht.

Eignung des KDD Prozesses

Auch wenn der KDD Prozess ursprünglich für einfache Daten und nicht für Complex Objects ausgelegt wurde, können solche mit ihm auch gut bearbeitet werden. Lediglich die Schritte der Vorverarbeitung und der Transformation weichen etwas von dem ursprünglichen Prozess ab.

Insbesondere durch geschickte Datentransformation lassen sich Complex Objekts mittels klassischer Data Mining Verfahren bearbeiten. Die zu betrachtenden Chargenbäume werden in dieser Arbeit mit geringem Aufwand auf eine einfachere Struktur, und zwar auf eine Liste der einfließenden Mengen reduziert. Dies ist nicht unbedingt immer möglich, sodass ggf. komplexere Verfahren zur Vorverarbeitung bzw. Bearbeitung von Daten herangezogen werden müssen. Das Vorgehen innerhalb des KDD Prozesses hängt somit stark von der eigentlichen Fragestellung ab.

Für das in dieser Arbeit betrachtete Problem ist der KDD Prozess geeignet und liefert potentiell nützliche Ergebnisse.

Eignung der Datenbasis

Bei der Datenbasis handelt es sich um die maschinell erzeugte Dokumentation eines Fertigungsprozesses. Obwohl es sich um maschinell generierte Daten handelt, wiesen diese partielle Defekte auf. Somit ist einiges an Aufbereitung notwendig, um die Daten in einen Zustand zu bringen, in welchem mit ihnen weiter gearbeitet werden kann. Insbesondere im Bereich der Aktivitäten ist einiges an Vorverarbeitung erforderlich.

Die Daten, welche im Data Mining Schritt bearbeitet wurden, wiesen eindeutig interne Strukturen auf, welche auf die Qualität der Fertigung schließen lassen. Somit scheinen die Daten trotz partieller Defekte für weiterführende Analysen geeignet.

Skalierbarkeit der Verfahren

Die hier verwendeten Verfahren kommen, abgesehen von der Datenvorverarbeitung, fast vollständig ohne menschliche Interaktion aus, wodurch der Mensch als limitierender Faktor fast vollständig entfällt. Auch sind Verfahren wie neuronale Netze für Parallelisierung auf unterschiedlicher Berechnungshardware geeignet [AAB+15].

Somit lassen sich auch große Mengen an Beispieldaten, welche für einige alternative Verfahren zu zahlreich wären, bearbeiten. Hierfür ist entsprechend Zeit oder Berechnungshardware notwendig. Doch auch Ansätze der neuronalen Netze stoßen bei immer größeren Feature Räumen, auch wenn später, an gewisse Grenzen.

4.2 Ausblick

Im Rahmen dieser Arbeit wird anhand der Chargenbaum Problematik der KDD Prozess für Complex Objects durchlaufen. Bei dieser ersten Betrachtung der Chargenbaum Daten wird das Konzept der Aktivitäten nicht betrachtet. Eben diese Aktivitäten werden zurzeit in einer unabhängigen Masterarbeit teilweise analysiert. Trotzdem gibt es immer noch die Problematik einer generalisierten Bearbeitungslösung für Aktivitäten, um Lösungswege auf unbekannte Aktivitäten zu übertragen. Eine solche Lösung wäre insbesondere bei mehreren Auftraggebern sinnvoll, da Aktivitäten zumindest teilweise kundenspezifisch sind, und daher stark individualisiert sein können.

Während des Data Mining Schrittes werden verschiedene einfache Verfahren betrachtet. Hierbei werden mit einfachsten Mitteln bereits deutliche Resultate erzielt. Über Anpassung

der Meta Parameter der Modelle könnten diese jedoch potenziell noch verbessert werden. Die Betrachtung anderer Verfahren, auch auf Baumstrukturen spezialisierte Verfahren wie beispielsweise in [RKBM10] beschrieben, würden sich anbieten.

Insbesondere in den Ansätzen der einfachen neuronalen Netze wie dem Perzeptron und dem Multi Layer Perzeptron besteht Potenzial für eine Verbesserung der Vorhersagegenauigkeit. Jedoch werden hierfür mehr Daten benötigt, welche aus einem direktem Datenbankexport entnommen werden könnten.

Neben einer Anpassung des Data Mining Schrittes wäre jedoch auch eine bessere Vorverarbeitung der Datenbasis sinnvoll. So sind hier ein Großteil der Defekte wie z. B. negative Transfermengen ignoriert worden. Unter Aufwendung von mehr Domänenwissen könnten die Daten potenziell sinnvoller aufbereitet und die Qualität der Ergebnisse erhöht werden. Einige Fehler in den Daten sind wahrscheinlich im Zuge der Datenselektion entstanden. Eine erneute Auswertung auf Basis eines direkten und ungefilterten Datenbankexports wäre sinnvoll, um ein realistisches Bild der Situation zu erlangen. Für eine kundenorientierte Durchführung muss ein klares Verfahren für den Datenaustausch und die Anonymisierung, falls diese firmenintern überhaupt notwendig ist, definiert werden.

Ein weiterer wichtiger Punkt ist das Einbeziehen des Kunden in den Schritt der Interpretation, da ohne Kommunikation mit diesem die Anwendung von objektiven Metriken von seinem Standpunkt aus eigentlich unmöglich ist. Hierzu würde sich beispielsweise eine Neumodellierung des KDD Prozesses unter Berücksichtigung von Kundenkommunikation eignen. Hierbei ist jedoch auch zu beachten, dass wie in Abschnitt 4.1 beschrieben, die Abläufe des KDD Prozesses je nach konkreter Aufgabenstellung voneinander abweichen können. Inwieweit eine vollständige Generalisierbarbeit des Prozesses bzw. eines konkreten Prozessframeworks möglich ist, muss sich noch zeigen.

Auch eine Weiterentwicklung des LSTM Ansatzes, um für nicht abgeschlossene Fertigungen Vorhersagen zu treffen, wäre denkbar, um etwa ein Frühwarnsystem, welches den Fertigungsprozess permanent überwacht, zu realisieren. Ein solches System könnte direkt bei einem Kunden in eine Fertigungsstrecke integriert werden. Neben einer Implementierung auf Basis von LSTMs ist auch eine Umsetzung mit klassischen Verfahren wie einfachen neuronalen Netzen unter entsprechender Datenvorverarbeitung denkbar.

Ein Frühwarnsystem wäre zwar ein potenziell für den Kunden nützlicher Mechanismus, für ihn wäre jedoch eine Ermittlung der Ursachen wesentlich attraktiver, um diese aktiv beseitigen zu können. Um eine eben solche Ursachenanalysen auf Basis von ML Verfahren durchführen zu können, müssen erklärbare Verfahren verwendet werden. Jedoch reicht eine Verwendung

von erklärbaren Verfahren nicht zwangsläufig aus, da sie teilweise recht unübersichtlich und konfus sind.

In dieser Arbeit wird an dem konkretem Beispiel eines biochemischen Fertigungsprozesses versucht, mit Mitteln des Data Minings Ergebnisse zu erzielen, welche den Anforderungen von Beierle für eine erfolgreiche Durchführung des KDD Prozesses genügen. Inwieweit die Ergebnisse jedoch auf andere Bereiche übertragbar sind, muss sich noch zeigen.

5 Messergebnisse

Sämtliche während der Experimente gesammelten Metriken werden in den Kapitel der Arbeit tabellarisch aufgelistet. Aus Platzgründen werden in den Bezeichnungen teilweise Abkürzungen verwendet, welche in der Tabelle 5.1 aufgeschlüsselt werden.

Tabelle 5.1: Abkürzungsschlüssel

Abkürzung	Bedeutung
TrvTe	Verhältnis von Trainings- zu Testdaten
MinErr	Minimaler aufgetretener Fehler
MaxErr	Maximaler aufgetretener Fehler
AvrErr	Durchschnittlicher Fehler
MinNegRec	Minimaler aufgetretener Recall der unterdurchschnittlichen Klasse
MaxNegRec	Maximaler aufgetretener Recall der unterdurchschnittlichen Klasse
AvrNegRec	Durchschnittlicher Recall der unterdurchschnittlichen Klasse
MinPosRec	Minimaler aufgetretener Recall der überdurchschnittlichen Klasse
MaxPosRec	Maximaler aufgetretener Recall der überdurchschnittlich Klasse
AvrPosRec	Durchschnittlicher Recall der überdurchschnittlichen Klasse
BesNegRec	Recall der unterdurchschnittlichen Klasse des Modells mit dem geringstem Fehler
BesPosRec	Recall der überdurchschnittlichen Klasse des Modells mit dem geringstem Fehler

Tabelle 5.2: C4.5, naiver Ansatz

Train-/Testdata	Error	Unterdurchschnitts Recall	Überdurchschnitts Recall
10/90	56.67	0.39	0.47
20/80	30.18	0.68	0.71
30/70	42.55	0.45	0.68
40/60	35.00	0.37	0.90
50/50	21.21	0.67	0.89
60/40	18.51	0.77	0.86
70/30	25.00	0.67	0.82
80/20	28.57	0.71	0.71
90/10	28.57	1.0	0.5

Tabelle 5.3: C4.5, naiver Ansatz, drei Buckets

Train-/Testdata	Error	Unterdurchschnitts Recall	Überdurchschnitts Recall
10/90	53.33	0.6428	0.3125
20/80	35.85	0.76	0.54
30/70	36.17	0.55	0.72
40/60	16.22	0.69	0.95
50/50	16.67	0.75	0.89
60/40	4.76	0.86	1.0
70/30	15.00	0.67	1.0
80/20	7.14	1.0	0.86
90/10	0.00	1.0	1.0

Tabelle 5.4: C4.5, naiver Ansatz, fünf Buckets

Train-/Testdata	Error	Unterdurchschnitts Recall	Überdurchschnitts Recall
10/90	35.19	0.60	0.69
20/80	37.50	0.65	0.61
30/70	26.19	0.55	0.91
40/60	28.21	0.50	0.90
50/50	31.25	0.57	0.78
60/40	8.00	0.91	0.93
70/30	13.33	0.75	0.91
80/20	15.38	0.83	0.86
90/10	0.00	1.00	1.00

Tabelle 5.5: C4.5, nicht naiver Ansatz

Train-/Testdata	Error	Unterdurchschnitts Recall	Überdurchschnitts Recall
10/90	43.33	0.21	0.88
20/80	30.19	0.68	0.71
30/70	44.68	0.41	0.68
40/60	27.50	0.53	0.90
50/50	21.21	0.67	0.89
60/40	22.22	0.77	0.79
70/30	15.00	0.78	0.91
80/20	21.43	0.86	0.71
90/10	28.57	1.0	0.5

Tabelle 5.6: C4.5, nicht naiver Ansatz, drei Buckets

Train-/Testdata	Error	Unterdurchschnitts Recall	Überdurchschnitts Recall
10/90	53.33	0.64	0.31
20/80	35.85	0.76	0.54
30/70	36.17	0.54	0.72
40/60	18.92	0.69	0.90
50/50	16.67	0.75	0.89
60/40	16.67	0.6	1.0
70/30	15.00	0.67	1.0
80/20	7.14	1.0	0.86
90/10	0.00	1.0	1.0

Tabelle 5.7: C4.5, nicht naiver Ansatz, fünf Buckets

Train-/Testdata	Error	Unterdurchschnitts Recall	Überdurchschnitts Recall
10/90	35.19	0.60	0.69
20/80	37.50	0.65	0.61
30/70	54.55	0.32	0.56
40/60	26.92	0.63	0.90
50/50	12.90	0.85	0.89
60/40	12.00	0.91	0.86
70/30	20.00	0.75	0.82
80/20	23.08	0.83	0.71
90/10	0.00	1.0	1.0

Tabelle 5.8: Ergebnisse Perzeptron, normalisiert, naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	35.135	50.450	42.396	0.088	0.632	0.425	0.463	0.963	0.735	0.474	0.833
20/80	33.333	47.475	38.857	0.157	0.765	0.633	0.417	0.938	0.589	0.706	0.625
30/70	22.093	45.349	29.260	0.250	0.977	0.723	0.190	0.929	0.691	0.750	0.810
40/60	24.324	43.243	31.059	0.237	0.763	0.694	0.389	0.917	0.684	0.737	0.778
50/50	27.869	42.623	32.315	0.281	0.844	0.744	0.345	0.897	0.603	0.844	0.586
60/40	20.408	42.857	30.229	0.280	0.880	0.706	0.375	0.875	0.689	0.800	0.792
70/30	24.324	43.243	30.886	0.158	0.789	0.682	0.444	1.000	0.701	0.684	0.833
80/20	20.833	41.667	32.817	0.231	0.846	0.641	0.364	1.000	0.708	0.846	0.727
90/10	25.000	41.667	27.067	0.333	0.667	0.641	0.500	1.000	0.817	0.667	0.833

Tabelle 5.9: Ergebnisse Perzeptron, standardisiert, naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	35.135	55.856	42.915	0.263	0.667	0.489	0.370	0.907	0.657	0.561	0.741
20/80	31.313	52.525	44.162	0.392	0.745	0.544	0.333	0.771	0.574	0.745	0.625
30/70	22.093	55.814	34.558	0.432	0.932	0.640	0.357	0.881	0.670	0.795	0.762
40/60	24.324	51.351	35.124	0.342	0.763	0.626	0.472	0.861	0.673	0.763	0.750
50/50	22.951	59.016	35.797	0.438	0.844	0.715	0.379	0.828	0.561	0.750	0.793
60/40	20.408	59.184	31.935	0.400	0.880	0.683	0.333	0.833	0.678	0.800	0.792
70/30	18.919	59.459	32.119	0.263	0.947	0.645	0.278	0.944	0.715	0.684	0.944
80/20	16.667	54.167	33.150	0.385	0.846	0.632	0.364	0.909	0.712	0.846	0.818
90/10	16.667	58.333	32.567	0.333	0.833	0.622	0.333	0.833	0.727	0.833	0.833

Tabelle 5.10: Ergebnisse Perzeptron, normalisiert, nicht naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	27.119	55.932	40.237	0.000	1.000	0.732	0.148	0.963	0.438	1.000	0.407
20/80	19.231	46.154	29.146	0.750	1.000	0.868	0.083	0.750	0.522	0.893	0.708
30/70	17.391	43.478	27.513	0.760	1.000	0.964	0.048	0.714	0.441	0.920	0.714
40/60	17.949	43.590	25.190	0.857	1.000	0.924	0.111	0.667	0.543	0.952	0.667
50/50	12.500	56.250	18.712	0.556	1.000	0.982	0.286	0.786	0.596	1.000	0.714
60/40	11.538	34.615	18.985	0.929	1.000	0.995	0.250	0.750	0.595	1.000	0.750
70/30	15.789	31.579	18.589	0.900	1.000	0.992	0.333	0.778	0.616	1.000	0.667
80/20	7.692	53.846	20.092	0.000	1.000	0.934	0.429	1.000	0.683	1.000	0.857
90/10	0.000	50.000	16.533	0.000	1.000	0.972	0.667	1.000	0.697	1.000	1.000

Tabelle 5.11: Ergebnisse Perzeptron, standardisiert, nicht naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	22.034	67.797	44.549	0.125	0.938	0.583	0.185	0.889	0.521	0.688	0.889
20/80	15.385	57.692	32.377	0.286	0.964	0.730	0.167	0.917	0.613	0.857	0.833
30/70	13.043	52.174	29.739	0.400	1.000	0.830	0.286	0.952	0.551	0.800	0.952
40/60	10.256	43.590	24.841	0.476	1.000	0.919	0.278	0.833	0.556	0.952	0.833
50/50	6.250	50.000	22.275	0.500	1.000	0.883	0.357	0.857	0.641	1.000	0.857
60/40	0.000	50.000	20.938	0.357	1.000	0.871	0.333	1.000	0.697	1.000	1.000
70/30	0.000	47.368	20.000	0.500	1.000	0.879	0.222	1.000	0.712	1.000	1.000
80/20	0.000	53.846	17.815	0.333	1.000	0.830	0.429	1.000	0.815	1.000	1.000
90/10	0.000	66.667	17.533	0.333	1.000	0.839	0.000	1.000	0.811	1.000	1.000

Tabelle 5.12: Ergebnisse Multi Layer Perzeptron (eine Hidden Layer mit Breite 100), normalisiert, naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	29.730	51.351	38.577	0.000	0.614	0.461	0.611	1.000	0.776	0.544	0.870
20/80	36.364	50.505	41.539	0.059	1.000	0.526	0.000	1.000	0.647	0.529	0.750
30/70	20.930	48.837	26.233	0.477	1.000	0.798	0.000	0.857	0.675	0.864	0.714
40/60	24.324	43.243	28.281	0.237	1.000	0.729	0.250	0.917	0.705	0.763	0.750
50/50	24.590	42.623	31.718	0.281	0.938	0.788	0.276	0.897	0.567	0.844	0.655
60/40	22.449	51.020	27.935	0.000	1.000	0.815	0.000	1.000	0.623	0.880	0.667
70/30	18.919	51.351	26.476	0.000	1.000	0.770	0.000	1.000	0.699	0.789	0.833
80/20	16.667	54.167	24.367	0.000	1.000	0.748	0.000	1.000	0.766	0.846	0.818
90/10	25.000	50.000	29.433	0.000	1.000	0.763	0.000	1.000	0.649	0.833	0.667

Tabelle 5.13: Ergebnisse Multi Layer Perzeptron (eine Hidden Layer mit Breite 100), standardisiert, naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	32.432	52.252	42.364	0.175	0.912	0.399	0.185	0.944	0.764	0.561	0.796
20/80	38.384	51.515	44.537	0.000	0.980	0.481	0.000	1.000	0.632	0.510	0.729
30/70	22.093	43.023	29.549	0.318	1.000	0.733	0.238	0.905	0.675	0.818	0.738
40/60	22.973	43.243	30.416	0.237	1.000	0.680	0.194	0.917	0.712	0.763	0.778
50/50	24.590	52.459	33.049	0.000	0.969	0.709	0.138	1.000	0.626	0.812	0.690
60/40	18.367	44.898	28.408	0.320	0.960	0.781	0.292	0.917	0.648	0.880	0.750
70/30	16.216	43.243	26.011	0.316	1.000	0.731	0.333	0.944	0.749	0.895	0.778
80/20	12.500	45.833	24.800	0.154	1.000	0.701	0.545	1.000	0.812	0.923	0.818
90/10	25.000	58.333	30.500	0.167	1.000	0.726	0.167	0.833	0.664	0.833	0.667

Tabelle 5.14: Ergebnisse Multi Layer Perzeptron (eine Hidden Layer mit Breite 100), normalisiert, nicht naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	27.119	55.932	40.061	0.000	1.000	0.703	0.000	1.000	0.477	0.969	0.444
20/80	21.154	57.692	32.208	0.000	1.000	0.870	0.000	1.000	0.453	0.857	0.708
30/70	19.565	54.348	30.104	0.000	1.000	0.988	0.000	1.000	0.355	1.000	0.571
40/60	17.949	46.154	23.426	0.905	1.000	0.973	0.000	0.667	0.524	1.000	0.611
50/50	9.375	56.250	21.337	0.000	1.000	0.943	0.000	1.000	0.585	1.000	0.786
60/40	7.692	53.846	17.692	0.000	1.000	0.957	0.000	1.000	0.667	1.000	0.833
70/30	10.526	52.632	19.368	0.000	1.000	0.957	0.000	1.000	0.639	1.000	0.778
80/20	7.692	53.846	19.815	0.000	1.000	0.890	0.000	1.000	0.726	1.000	0.857
90/10	0.000	50.000	15.067	0.000	1.000	0.964	0.000	1.000	0.735	1.000	1.000

Tabelle 5.15: Ergebnisse Multi Layer Perzeptron (eine Hidden Layer mit Breite 100), standardisiert, nicht naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	27.119	66.102	44.359	0.188	1.000	0.683	0.037	0.889	0.406	0.812	0.630
20/80	17.308	53.846	29.362	0.000	1.000	0.833	0.042	1.000	0.559	0.893	0.750
30/70	17.391	45.652	29.017	0.600	1.000	0.906	0.143	1.000	0.476	0.960	0.667
40/60	7.692	41.026	24.308	0.762	1.000	0.961	0.167	0.833	0.518	1.000	0.833
50/50	6.250	37.500	17.738	0.556	1.000	0.936	0.214	0.857	0.676	1.000	0.857
60/40	0.000	34.615	14.923	0.571	1.000	0.931	0.417	1.000	0.757	1.000	1.000
70/30	0.000	31.579	15.032	0.700	1.000	0.936	0.333	1.000	0.753	1.000	1.000
80/20	0.000	38.462	12.615	0.500	1.000	0.877	0.429	1.000	0.871	1.000	1.000
90/10	0.000	33.333	8.267	0.667	1.000	0.905	0.333	1.000	0.929	1.000	1.000

Tabelle 5.16: Ergebnisse Multi Layer Perzeptron (zwei Hidden Layer mit Breite 100), normalisiert, naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	31.532	51.351	41.095	0.000	0.684	0.423	0.556	1.000	0.765	0.579	0.796
20/80	35.354	51.515	42.578	0.000	1.000	0.511	0.000	1.000	0.641	0.529	0.771
30/70	19.767	51.163	26.484	0.000	1.000	0.802	0.000	1.000	0.665	0.864	0.738
40/60	24.324	51.351	28.314	0.000	1.000	0.731	0.000	1.000	0.702	0.763	0.750
50/50	22.951	52.459	32.170	0.000	1.000	0.789	0.000	1.000	0.556	0.844	0.690
60/40	18.367	51.020	28.155	0.000	1.000	0.817	0.000	1.000	0.615	0.880	0.750
70/30	18.919	51.351	26.486	0.000	1.000	0.752	0.000	1.000	0.717	0.842	0.778
80/20	12.500	54.167	24.583	0.000	1.000	0.729	0.000	1.000	0.784	0.923	0.818
90/10	16.667	50.000	29.400	0.000	1.000	0.741	0.000	1.000	0.671	1.000	0.667

Tabelle 5.17: Ergebnisse Multi Layer Perzeptron (zwei Hidden Layer mit Breite 100), standardisiert, naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	34.234	51.351	42.238	0.000	1.000	0.370	0.000	1.000	0.796	0.509	0.815
20/80	37.374	51.515	42.727	0.000	0.686	0.492	0.458	1.000	0.659	0.667	0.583
30/70	22.093	51.163	27.893	0.000	1.000	0.781	0.000	1.000	0.659	0.841	0.714
40/60	20.270	51.351	28.811	0.000	0.816	0.708	0.389	1.000	0.716	0.763	0.833
50/50	21.311	54.098	31.816	0.000	1.000	0.718	0.207	1.000	0.642	0.844	0.724
60/40	16.327	51.020	26.514	0.000	0.960	0.803	0.292	1.000	0.664	0.880	0.792
70/30	13.514	48.649	24.951	0.263	1.000	0.720	0.000	1.000	0.783	0.789	0.944
80/20	12.500	54.167	24.900	0.000	1.000	0.695	0.000	1.000	0.817	0.846	0.909
90/10	16.667	50.000	31.433	0.000	1.000	0.699	0.000	1.000	0.673	0.833	0.833

Tabelle 5.18: Ergebnisse Multi Layer Perzeptron (zwei Hidden Layer mit Breite 100), normalisiert, nicht naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	28.814	54.237	39.885	0.000	1.000	0.699	0.000	1.000	0.485	1.000	0.370
20/80	21.154	53.846	34.262	0.000	1.000	0.846	0.000	1.000	0.437	0.857	0.708
30/70	21.739	54.348	31.348	0.000	1.000	0.974	0.000	1.000	0.345	1.000	0.524
40/60	15.385	53.846	24.574	0.000	1.000	0.946	0.000	1.000	0.530	1.000	0.667
50/50	9.375	56.250	21.887	0.000	1.000	0.946	0.000	1.000	0.569	1.000	0.786
60/40	7.692	53.846	18.154	0.000	1.000	0.965	0.000	1.000	0.648	1.000	0.833
70/30	10.526	52.632	18.126	0.000	1.000	0.968	0.000	1.000	0.653	1.000	0.778
80/20	0.000	53.846	14.308	0.000	1.000	0.917	0.000	1.000	0.806	1.000	1.000
90/10	0.000	50.000	9.933	0.000	1.000	0.979	0.000	1.000	0.823	1.000	1.000

Tabelle 5.19: Ergebnisse Multi Layer Perzeptron (zwei Hidden Layer mit Breite 100), standardisiert, nicht naiver Ansatz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	32.203	61.017	45.797	0.000	1.000	0.677	0.000	1.000	0.382	0.750	0.593
20/80	11.538	46.154	29.654	0.536	1.000	0.854	0.000	0.917	0.528	0.893	0.875
30/70	19.565	54.348	28.617	0.000	1.000	0.917	0.000	1.000	0.472	0.960	0.619
40/60	7.692	53.846	24.810	0.000	1.000	0.953	0.000	1.000	0.518	1.000	0.833
50/50	6.250	43.750	17.562	0.611	1.000	0.939	0.000	0.929	0.677	0.944	0.929
60/40	3.846	53.846	15.877	0.000	1.000	0.934	0.000	1.000	0.733	1.000	0.917
70/30	0.000	52.632	16.695	0.000	1.000	0.911	0.000	1.000	0.746	1.000	1.000
80/20	0.000	53.846	13.077	0.000	1.000	0.871	0.000	1.000	0.867	1.000	1.000
90/10	0.000	50.000	8.467	0.000	1.000	0.897	0.000	1.000	0.933	1.000	1.000

Tabelle 5.20: Ergebnisse LSTM, Breitensuche, normalisiert

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	45.763	45.763	45.763	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
20/80	30.769	46.154	42.308	0.964	1.000	0.991	0.000	0.375	0.094	0.964	0.375
30/70	45.652	45.652	45.652	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
40/60	46.154	46.154	46.154	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
50/50	43.750	43.750	43.750	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
60/40	46.154	46.154	46.154	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
70/30	47.368	47.368	47.368	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
80/20	53.846	53.846	53.846	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
90/10	50.000	50.000	50.000	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000

Tabelle 5.21: Ergebnisse LSTM, Breitensuche, normalisiert, umgekehrte Sequenz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	23.729	47.458	33.898	0.906	0.938	0.922	0.037	0.593	0.352	0.906	0.593
20/80	34.615	46.154	40.385	0.607	0.964	0.795	0.042	0.583	0.365	0.929	0.333
30/70	32.609	47.826	39.130	0.240	0.840	0.640	0.143	0.952	0.571	0.760	0.571
40/60	23.077	46.154	39.103	0.905	0.952	0.917	0.056	0.611	0.250	0.905	0.611
50/50	37.500	43.750	41.406	0.889	1.000	0.931	0.000	0.286	0.143	0.889	0.286
60/40	42.308	46.154	45.192	0.929	1.000	0.946	0.083	0.083	0.083	1.000	0.083
70/30	42.105	52.632	48.684	0.000	0.900	0.450	0.222	1.000	0.583	0.900	0.222
80/20	53.846	61.538	55.769	0.667	0.833	0.792	0.143	0.143	0.143	0.833	0.143
90/10	50.000	66.667	54.167	0.000	0.667	0.333	0.333	1.000	0.583	0.000	1.000

Tabelle 5.22: Ergebnisse LSTM, Tiefensuche, normalisiert

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	30.508	45.763	36.864	0.719	1.000	0.852	0.000	0.556	0.370	0.969	0.370
20/80	26.923	34.615	30.288	0.786	0.964	0.884	0.375	0.542	0.479	0.893	0.542
30/70	30.435	30.435	30.435	0.800	1.000	0.950	0.333	0.571	0.393	1.000	0.333
40/60	28.205	30.769	30.128	1.000	1.000	1.000	0.333	0.389	0.347	1.000	0.389
50/50	28.125	43.750	32.031	1.000	1.000	1.000	0.000	0.357	0.268	1.000	0.357
60/40	26.923	26.923	26.923	1.000	1.000	1.000	0.417	0.417	0.417	1.000	0.417
70/30	31.579	36.842	32.895	0.400	1.000	0.850	0.333	0.889	0.472	1.000	0.333
80/20	30.769	53.846	40.385	0.000	1.000	0.750	0.000	1.000	0.464	1.000	0.429
90/10	33.333	50.000	41.667	1.000	1.000	1.000	0.000	0.333	0.167	1.000	0.333

Tabelle 5.23: Ergebnisse LSTM, Tiefensuche, normalisiert, umgekehrte Sequenz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	32.203	47.458	39.831	0.656	0.938	0.812	0.037	0.704	0.352	0.656	0.704
20/80	34.615	50.000	44.712	0.679	0.964	0.848	0.042	0.625	0.208	0.679	0.625
30/70	39.130	47.826	44.565	0.600	0.920	0.820	0.095	0.619	0.238	0.600	0.619
40/60	43.590	46.154	44.231	0.905	0.952	0.929	0.056	0.167	0.125	0.905	0.167
50/50	21.875	43.750	38.281	0.833	0.944	0.889	0.071	0.714	0.268	0.833	0.714
60/40	42.308	46.154	43.269	0.571	0.929	0.839	0.167	0.500	0.250	0.929	0.167
70/30	47.368	52.632	48.684	0.100	0.900	0.675	0.111	0.889	0.333	0.900	0.111
80/20	53.846	53.846	53.846	0.667	1.000	0.833	0.000	0.286	0.143	0.667	0.286
90/10	50.000	66.667	54.167	0.000	1.000	0.583	0.000	0.667	0.333	0.667	0.333

Tabelle 5.24: Ergebnisse LSTM, Breitensuche, standardisiert

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	28.814	45.763	37.712	0.969	1.000	0.984	0.000	0.407	0.194	0.969	0.407
20/80	28.846	46.154	34.135	0.964	1.000	0.973	0.000	0.417	0.292	0.964	0.417
30/70	30.435	45.652	41.848	1.000	1.000	1.000	0.000	0.333	0.083	1.000	0.333
40/60	46.154	46.154	46.154	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
50/50	43.750	43.750	43.750	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
60/40	46.154	46.154	46.154	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
70/30	47.368	47.368	47.368	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000
80/30	46.154	53.846	51.923	0.000	1.000	0.750	0.000	1.000	0.250	0.000	1.000
90/10	50.000	50.000	50.000	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000

Tabelle 5.25: Ergebnisse LSTM, Breitensuche, standardisiert, umgekehrte Sequenz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	27.119	50.847	37.712	0.719	0.969	0.867	0.111	0.741	0.333	0.719	0.741
20/80	23.077	50.000	42.308	0.536	0.964	0.750	0.042	0.625	0.375	0.893	0.625
30/70	39.130	58.696	45.109	0.040	0.880	0.520	0.190	0.857	0.583	0.520	0.714
40/60	43.590	48.718	46.154	0.667	0.952	0.869	0.056	0.333	0.153	0.905	0.167
50/50	34.375	43.750	39.844	0.833	0.944	0.875	0.143	0.357	0.250	0.889	0.357
60/40	42.308	57.692	47.115	0.214	0.929	0.732	0.083	0.667	0.292	0.857	0.250
70/30	47.368	47.368	47.368	0.800	0.900	0.875	0.111	0.222	0.139	0.800	0.222
80/20	46.154	53.846	51.923	0.167	0.833	0.667	0.143	0.857	0.321	0.167	0.857
90/10	50.000	66.667	54.167	0.000	0.667	0.500	0.333	0.667	0.417	0.667	0.333

Tabelle 5.26: Ergebnisse LSTM, Tiefensuche, standardisiert

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	33.898	40.678	37.288	0.594	0.719	0.656	0.556	0.630	0.593	0.719	0.593
20/80	28.846	28.846	28.846	0.893	0.964	0.911	0.417	0.500	0.479	0.893	0.500
30/70	28.261	30.435	29.891	0.920	1.000	0.980	0.333	0.476	0.369	0.920	0.476
40/60	30.769	35.897	32.692	0.619	1.000	0.786	0.333	0.722	0.542	0.905	0.444
50/50	28.125	46.875	35.938	0.556	1.000	0.806	0.357	0.643	0.429	1.000	0.357
60/40	26.923	38.462	29.808	0.571	1.000	0.893	0.417	0.667	0.479	1.000	0.417
70/30	31.579	36.842	32.895	0.600	1.000	0.900	0.333	0.667	0.417	1.000	0.333
80/20	30.769	30.769	30.769	1.000	1.000	1.000	0.429	0.429	0.429	1.000	0.429
90/10	33.333	33.333	33.333	1.000	1.000	1.000	0.333	0.333	0.333	1.000	0.333

Tabelle 5.27: Ergebnisse LSTM, Tiefensuche, standardisiert, umgekehrte Sequenz

TrvTe	MinErr	MaxErr	AvrErr	MinNegRec	MaxNegRec	AvrNegRec	MinPosRec	MaxPosRec	AvrPosRec	BesNegRec	BesPosRec
10/90	27.119	38.983	33.475	0.750	1.000	0.883	0.148	0.704	0.407	0.750	0.704
20/80	23.077	38.462	31.731	0.821	0.929	0.866	0.375	0.667	0.469	0.857	0.667
30/70	28.261	45.652	38.587	0.720	0.920	0.830	0.095	0.714	0.357	0.720	0.714
40/60	20.513	46.154	37.821	0.905	0.952	0.917	0.111	0.611	0.278	0.952	0.611
50/50	31.250	56.250	42.188	0.056	0.944	0.667	0.143	0.929	0.464	0.833	0.500
60/40	26.923	42.308	38.462	0.500	0.929	0.804	0.167	1.000	0.396	0.500	1.000
70/30	47.368	47.368	47.368	0.800	0.900	0.875	0.111	0.222	0.139	0.900	0.111
80/20	53.846	53.846	53.846	0.667	1.000	0.833	0.000	0.286	0.143	0.833	0.143
90/10	50.000	66.667	54.167	0.000	0.667	0.500	0.333	0.667	0.417	0.667	0.333

6 Anhang

6.1 Tabellenstrukturen

6.1.1 Struktur von Baum_Kern

Die folgende Auflistung beschreibt die Bedeutungen der einzelnen Attribute.

- Das Attribut **ID** war in den ursprünglichen Daten nicht vorhanden, und wurde eingeführt, da kein eindeutiger Primärschlüssel erkennbar ist, um einen Datensatz über ein Attribut eindeutig identifizieren zu können.
- Das Attribut **FerAufNumFlag** (Fertigungsauftrags Nummer Flag) beschreibt, ob es sich bei einem Fertigungsprozess um einen normalen Auftrag (Normal) oder um einen Auftrag zur Reinigung (Reinig) handelt. Es sind auch andere Freitexte möglich, jedoch sind in der Datenbasis keine anderen vertreten.
- Die **QuellMatBez** (Quell Material Bezeichnung) ist die textuelle Bezeichnung des Quellmaterials, welches zu einer gewissen Menge in ein Zielmaterial einfließt.
- Die **QuellMatNum** (Quell Material Nummer) ist eine Möglichkeit, ein Material eindeutig zu identifizieren, es handelt sich auf den ersten Blick um ein rein numerisches Attribut, jedoch können beliebige Zeichenfolgen an dieser Stelle verwendet werden.
- Um ein konkretes Material eindeutig identifizieren zu können, wird die **QuellChaNum** (Quell Chargen Nummer) benötigt. Diese beschreibt die Fertigungs-Charge, aus welcher ein konkretes Material stammt und ist nur in Kombination mit der Materialnummer eindeutig. Genau wie bei einer Materialnummer auch handelt es sich um ein Freitextfeld.
- Über die **QuellFerAufNum** (Quell Fertigungs Auftrags Nummer) kann ermittelt werden, zu welchem konkretem Fertigungsauftrag das Quellmaterial gehört. Mit dieser Information kann jedoch nicht viel angefangen werden, da hierzu Daten aus einer Tabelle erforderlich wären, welche leider nicht bereit gestellt wurde. Somit lässt sich über dieses

Feld lediglich ermitteln, ob zwei konkrete Materialien aus dem selben Fertigungsauftrag stammen. Wieder handelt es sich um ein Freitextfeld.

- Bei der **QuellMenge** (Quell Menge) handelt es sich um die Menge, mit welcher das Quellmaterial in das Zielmaterial einfließt. An dieser Stelle kann ein beliebiger Float Wert stehen.
- Die **QuellMengeEinheit** (Quell Mengen Einheit) definiert die Einheit der QuellMenge. Bei diesem Feld handelt es sich ebenfalls um ein Freitextfeld.
- Die **QuellLauNum** (Quell Lauf Nummer) identifiziert einen sogenannten Lauf unter den Fertigungsprozessen, in diesem Fall ist der Lauf der Quelle gemeint.
- Bei der **ZielMatNum** (Ziel Material Nummer) handelt es sich wie bei der QuellMatNum um eine Materialnummer, diese identifiziert jedoch das Zielmaterial, es gelten die selben Regeln wie auch bei der QuellMatNum.
- Ebenso wie die QuellChaNum dient auch die **ZielCharNummer** (Ziel Chargen Nummer) der eindeutigen Identifizierung eines konkreten Materials, jedoch der eindeutigen Identifizierung des Zielmaterials. Es gelten die selben Regeln wie für die QuellChaNum.
- Die **ZielFertAufNum** (Ziel Fertigungs Auftrags Nummer) beschreibt, zu welchem Fertigungsauftrag das konkrete Zielmaterial gehört. Es gelten leider sie selben Limitierungen wie für QuellFerAufNum.
- Die **ZielLauNum** (Ziel Lauf Nummer) ist das Gegenstück zur Quell Lauf Nummer und identifiziert ebenfalls einen sogenannten Lauf unter den Fertigungsprozessen, jedoch beschreibt sie in diesem Fall den Ziellauf.

6.1.2 Struktur der FertigungsZiel Tabelle

Die folgende Auflistung schlüsselt die Bedeutung aller Attribute der FertigungsZiel Tabelle auf.

- **ID** Dieses Attribut war in den ursprünglichen Daten nicht vorhanden, und wurde eingeführt, da kein eindeutiger Primärschlüssel vorhanden war.
- In der Fertigung gibt es ein Konzept, welches als Lauf bezeichnet wird, ein solcher wird über die **AARF_LauNum** (Laufnummer) eindeutig identifiziert.

- Ebenso wie bei der Baum_Kern tabelle kann über die **AARF_FerAufNum** (Fertigungsauftragsnummer) ermittelt werden, zu welchem konkretem Fertigungsauftrag das Quellmaterial gehört. Jedoch ist diese Information nicht sonderlich hilfreich, da keinerlei Informationen über die Fertigungsaufträge vorliegen. Und somit lässt sich ebenfalls nur ermitteln, ob zwei konkrete Materialien aus dem selben Fertigungsauftrag stammen. Es handelt sich um ein Freitextfeld.

- Das Feld **AERI_FurChaNum** (Für Chargennummer) enthält die Chargennummer des konkreten, produzierten Materials. Auch hier kann ein beliebiger Text verwendet werden.

- **AARF_MatNum** (Materialnummer) beschreibt das konkrete Material, welches als Ziel einer Fertigung angestrebt wurde. In den vorliegenden Daten ist lediglich eine Ausprägung vorhanden.

- Die Qualität des Endproduktes ist in **AERI_FloWer** (Float Wert) abgelegt. Konkret ist hier die Anzahl der Zellen pro Volumeneinheit in der resultierenden Suspension gespeichert.

- Das **IstEin_AERI_Dat** (Ist Einstellungs-Datum) beschreibt den Tag, an welchem der Eintrag in dieser Tabelle getätigt wurde.

6.1.3 Aktivitaeten Tabelle

Einige Felder aus der Aktivitaeten Tabelle werden hier beschrieben, jedoch konnte leider nicht für alle Attribute die Bedeutung eindeutig geklärt werden.

- Wie auch bei den anderen Tabellen wurde die Spalte **ID** künstlich hinzugefügt, um eine Zeile in der Tabelle möglichst einfach eindeutig identifizieren zu können.

- Um eine konkrete Art von Aktivität eindeutig zu identifizieren, ist das Feld **AKRA_AktNum** (Aktivitätennummer) geeignet, es enthält einen beliebigen Alpha numerischen Wert, welcher eine Art von Aktivität eindeutig identifiziert. Jedoch ist es nicht möglich, weitere Informationen über diesen Wert zu erhalten, da die Tabelle, welche weitere Informationen enthalten würde, nicht mitgeliefert wurde.

- Neben der Aktivitätennummer wird auch eine **AKRA_AktBez** (Aktivitätenbezeichnung) abgelegt, welche für eine Aktivitätennummer immer identisch ist. Beispielhafte Werte wären etwa 'Filter einbauen: Be- und Entlüftungsfilter' oder 'Identifikation Medienleitung'.

- Um zu erkennen, um was für eine Art von Datensatz es sich handelt, muss **AKRA_AktTyp** (Aktivitäten Typ) zur Hilfe genommen werden, an dieser Stelle ist ein Freitext möglich, welcher die Art der gespeicherten Daten aufschlüsselt. Werte, welche beispielsweise in den bereitgestellten Daten auftreten, sind: 'Objekt', 'Text', 'Attributiv', 'Messwert' oder 'Liste'. Diese Aufzählung ist und kann nicht erschöpft sein, da beliebige neue Werte deklariert werden können.
- Eine Aktivität entsteht immer im Zuge einer Arbeitsanweisung, welche über die **AWRA_ArbAnwNum** (Arbeitsanweisungs Nummer) identifiziert werden kann. Leider liegen hierzu neben der Bezeichnung der Anweisung ebenfalls keine weiteren Informationen vor.
- Die **AWRA_ArbAnwBez** (Arbeitsanweisungsbezeichnung) ist der Name bzw. die Bezeichnung für die Arbeitsanweisung, als welcher ein Eintrag der Tabelle entstanden ist.
- Die Art einer Aktivität wird über das **AWRA_Typ** (Type) Feld definiert, aus welcher die Quelle eine Aktivität stammt. Eine erschöpfte Liste der möglichen Werte: 'Bestand Erzeugung', 'Einsatzmaterial', 'Manuell', 'Objektbearbeitung', 'Objekterzeugung', 'Objektzerlegung', 'PLS', 'Protokolle'.
- Obwohl die Laufnummer bereits im zugehörigen Eintrag der Baum_Kern Tabelle vorhanden ist, wird die **AARF_LauNum** (Laufnummer) an dieser Stelle redundant abgelegt.
- Die **AARF_FerAufNum** (Fertigungsauftragsnummer) wird ebenfalls redundant abgelegt.
- Die Zuordnung zu einem Eintrag der Baum_Kern Tabelle erfolgt über eine Kombination aus **AERI_FurChaNum** (Für Chargennummer) und AARF_MatNum.
- Die **AARF_MatNum** (Materialnummer) dient in Kombination mit der AERI_FurChaNum der Zuordnung zu einem konkretem Eintrag der Baum_Kern Tabelle.
- **AARF_MatBez** (Materialbezeichnung) wird redundant zu der Baum_Kern Tabelle gehalten.
- Eine Handlungsanweisung an einen Operator kann in unterschiedlichen Varianten vorliegen, die konkret zutreffende wird über **AARF_HerVorVar** (Herstellungsvorschrifts-Variante) dokumentiert.

- Neben unterschiedlichen Varianten kann eine Vorschrift auch in unterschiedlichen Versionen vorliegen, um also zurückverfolgen zu können, welche Version genau vorlag, muss diese in **AARF_HerVorVer** (Herstellungsvorschrifts-Version) gespeichert werden.
- Über den **AARF_IstSta** (Ist Start) wird je nach Art der Aktivität der Startpunkt eben dieser festgelegt.
- **AARF_IstEnd** (Ist Ende) beschreibt ebenso wie AARF_IstSta den zeitlichen Kontext der Aktivität.
- Eine Fertigung findet in verschiedenen, räumlich getrennten Bereichen statt, diese werden als Produktionseinheit bezeichnet, sie werden an den Aktivitäten in dem Feld **AARP_ProEin** (Produktionseinheit) abgelegt.
- Die für den durchführenden Mitarbeiter relevante **AGRA_ArbGanNum** (Arbeits- gangs-Nummer) wird in diesem Feld abgelegt.
- Zusätzlich zu einer Arbeitsgangs-Nummer wird auch die **AGRA_ArbGanBez** (Arbeitsgangs-Bezeichnung) redundant an dieser Stelle gespeichert.
- Der **AERS_SolBooWer** (Soll Boolean Wert) enthält einen Zielwert für die Durchführung der Aktivität, wie dieser jedoch mit einem anderen Zielwert als Boolean Wert Text zusammenhängt, ist nicht klar.
- Das Feld **AERI_FloWer** (Float Wert) hat je nach Aktivitätenart eine abweichende Bedeutung, sodass ihm keine eindeutige Bedeutung zugeschrieben werden kann.
- ebenso wie der Float Wert hat auch der **AERI_BooWerTex** (Boolean Wert Text) je nach Aktivitätenart eine abweichende Bedeutung, so das ihm keine eindeutige bedeutung zugeschrieben werden kann.
- Der **AERI_TexWer** (Text Wert) enthält häufig den Float Wert erweitert um eine Maßeinheit. Jedoch finden sich auch andere Nutzungen für dieses Attribut.
- Wie auch der Text Wert werden im **AERI_Tex** (Text) unterschiedliche Bedeutungen kodiert, so werden an dieser Stelle teilweise Bemerkungen oder die Namen der Bediensteten abgelegt.
- Das **AERI_DATUM** (Datum) Feld beschreibt den Start- oder Endzeitpunkt einer aufgetretenen Aktivität.

Literaturverzeichnis

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

[acm] *ACM Digital Library*. https://dl.acm.org/

[ADM+92] Atkinson, Malcolm ; DeWitt, David ; Maier, David ; Bancilhon, François ; Dittrich, Klaus ; Zdonik, Stanley: Building an Object-oriented Database System. Version: 1992. http://dl.acm.org/citation.cfm?id=140592.140595. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1992. – ISBN 1–55860–169–4, Kapitel The Object-oriented Database System Manifesto, 1–20

[BCD+09] Berthold, Michael R. ; Cebron, Nicolas ; Dill, Fabian ; Gabriel, Thomas R. ; Kötter, Tobias ; Meinl, Thorsten ; Ohl, Peter ; Thiel, Kilian ; Wiswedel, Bernd: KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. In: *SIGKDD Explor. Newsl.* 11 (2009), November, Nr. 1, 26–31. http://dx.doi.org/10.1145/1656274.1656280. – DOI 10.1145/1656274.1656280. – ISSN 1931–0145

[BI06] Beierle, Christoph ; Isberner, Gabriele K.: *Methoden wissensbasierter Systeme : Grundlagen, Algorithmen, Anwendungen*. 3., überarb. u. erw. Aufl. Braunschweig [u.a.] : Viewg, 2006

[CCK+00] Chapman, Pete ; Clinton, Julian ; Kerber, Randy ; Khabaza, Thomas ; Reinartz, Thomas ; Shearer, Colin ; Wirth, Rudiger: CRISP-DM 1.0 Step-by-step data mining guide / The CRISP-DM consortium. Version: August 2000. https://maestria-datamining-2010.googlecode.com/svn-history/r282/trunk/dmct-teorica/tp1/CRISPWP-0800.pdf. 2000. – Forschungsbericht

[CL16] Cleve, J. ; Lämmel, U.: *Data Mining*. De Gruyter, 2016 (De Gruyter Studium). https://books.google.de/books?id=LngDDgAAQBAJ. – ISBN 9783110456776

[CRI] *CRISP-DM, still the top methodology for analytics, data mining, or data science projects.* http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html

[Csá01] Csáji, Balázs C.: Approximation with artificial neural networks. (2001)

[dar] *Explainable Artificial Intelligence (XAI).* https://www.darpa.mil/program/explainable-artificial-intelligence

[DS93] DasGupta, Bhaskar ; Schnitger, Georg: The Power of Approximating: a Comparison of Activation Functions. Version: 1993. http://papers.nips.cc/paper/692-the-power-of-approximating-a-comparison-of-activation-functions.pdf. In: Hanson, S. J. (Hrsg.) ; Cowan, J. D. (Hrsg.) ; Giles, C. L. (Hrsg.): *Advances in Neural Information Processing Systems 5*. Morgan-Kaufmann, 1993, 615–622

[FPSS96] Fayyad, Usama M. ; Piatetsky-Shapiro, Gregory ; Smyth, Padhraic: Advances in Knowledge Discovery and Data Mining. Version: 1996. http://dl.acm.org/citation.cfm?id=257938.257942. Menlo Park, CA, USA : American Association for Artificial Intelligence, 1996. – ISBN 0–262–56097–6, Kapitel From Data Mining to Knowledge Discovery: An Overview, 1–34

[HHK15] Heer, Jeffrey ; Hellerstein, Joseph ; Kandel, Sean: Predictive Interaction for Data Transformation. In: *Conference on Innovative Data Systems Research (CIDR)*, 2015

[Hop82] Hopfield, J. J.: Neural networks and physical systems with emergent collective computational abilities. In: *Proceedings of the National Academy of Sciences of*

the United States of America 79 (1982), April, Nr. 8, 2554–2558. http://view.ncbi.nlm.nih.gov/pubmed/6953413]. – ISSN 0027–8424

[HS97] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Comput.* 9 (1997), November, Nr. 8, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735. – DOI 10.1162/neco.1997.9.8.1735. – ISSN 0899–7667

[KB14] Kingma, Diederik P. ; Ba, Jimmy: *Adam: A Method for Stochastic Optimization.* 2014

[MP43] Mcculloch, Warren ; Pitts, Walter: A Logical Calculus of Ideas Immanent in Nervous Activity. In: *Bulletin of Mathematical Biophysics* 5 (1943), S. 127–147

[MP69] Minsky, Marvin ; Papert, Seymour: *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA, USA : MIT Press, 1969

[PSM94] Piatetsky-Shapiro, G. ; Matheus, C. J.: The interestingness of deviations. In: *AAAI'94 Workshop on Knowledge Discovery in Databases*, AAAI Press, July 1994, S. 25–36

[Qui93] Quinlan, J. R.: *C4.5: Programs for Machine Learning.* San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1993. – ISBN 1–55860–238–0

[RKBM10] Rieck, Konrad ; Krueger, Tammo ; Brefeld, Ulf ; Müller, Klaus-Robert: Approximate Tree Kernels. In: *J. Mach. Learn. Res.* 11 (2010), März, 555–580. http://dl.acm.org/citation.cfm?id=1756006.1756022. – ISSN 1532–4435

[ST95] Silberschatz, Abraham ; Tuzhilin, Alexander: On Subjective Measures of Interestingness in Knowledge Discovery. In: Fayyad, Usama M. (Hrsg.) ; Uthurusamy, Ramasamy (Hrsg.): *KDD*, AAAI Press, 1995. – ISBN 0–929280–82–2, 275-281

[sta] *Neural Networks - History.* https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/index.html

[TSSP16] Tompson, Jonathan ; Schlachter, Kristofer ; Sprechmann, Pablo ; Perlin, Ken: Accelerating Eulerian Fluid Simulation With Convolutional Networks. In: *CoRR* abs/1607.03597 (2016). http://arxiv.org/abs/1607.03597

[WH96] Williams, Graham ; Huang, Zhexue: Modelling the KDD Process A Four Stage Process and Four Element Model. (1996), 07, S. 8

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 16. November 2017 Lukas Lühr