Data Mining mit Kalibrierdaten aus der Motorenentwicklung

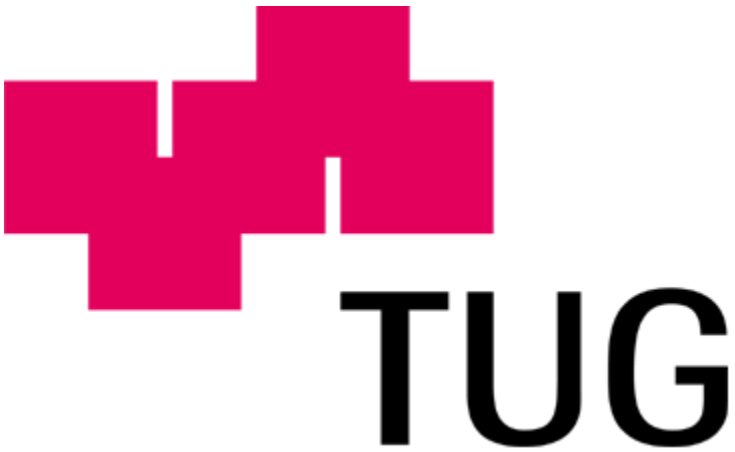
Masterarbeit
an der
Technischen Universität Graz

vorgelegt von
Stefan Thamerl

Mai 2014

Betreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Wotawa Franz

Institut für Softwaretechnologie(IST),
Technische Universität Graz
A-8010 Graz



Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)

Kurzfassung

Die Arbeit befasst sich mit einer realen Data Mining Problemstellung im betrieblichen Umfeld. Ziel ist die Implementierung eines Prototyps, der aufzeigt, dass Data Mining Methoden auf Kalibrierdaten der Motorenentwicklung angewandt werden können. Hierzu werden zuerst die wichtigsten Data Mining Prozessmodelle genannt, die eine Orientierungshilfe für eine derartige Problemstellung bieten. Das CRISP-DM Prozessmodell wird näher beschrieben, weil es auch für das Lösen dieser Problemstellung ausgewählt wurde. Einzelne Phasen dieses Prozessmodells werden anhand der gegebenen Problemstellung in der praktischen Umsetzung vorgeführt. Die Arbeit geht der Frage nach, welche Data Mining Methode sich gut für diese Datenbasis eignet. Die Assoziationsanalyse wurde in diesem Zusammenhang ausgewählt. Es werden zwei bekannte Algorithmen der Assoziationsanalyse, der Apriori und FP-growth beschrieben und miteinander verglichen. Anschließend werden noch Möglichkeiten der Weiterverarbeitung von Assoziationsregeln diskutiert. Im Besonderen wird hier auf die Möglichkeit der Ausreißererkennung in Transaktionsdaten eingegangen. Abschließend werden die Ergebnisse anhand realer Kalibrierdaten evaluiert und der implementierte Prototyp vorgestellt.

Abstract

This diploma thesis deals with a real data mining problem in a business environment. The goal is the development of a prototype which proves that data mining methods can be used with calibration data of engine development. First of all some data mining process models are shown, because they are a proper guide for such a task. The CRISP-DM process model is described in detail since it was chosen for this task. Some individual steps of this process model are implemented in practical usage for the prototype. This work tries to give an answer to the question which data mining method is usable for this kind of data. Furthermore the association analysis is delineated and two algorithms of this method, the Apriori and FP-growth are described and compared. Afterwards some possibilities of further processing for association rules are presented, in particular the outlier detection in transactional data. Finally there is an evaluation of the prototype with real calibration data and the implemented prototype will be presented.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die durch ihre fachliche sowie persönliche Unterstützung zum Gelingen dieser Arbeit beigetragen haben.

Eingangs möchte ich mich bei meinem Betreuer Univ.-Prof. Dipl.-Ing. Dr. techn. Franz Wotawa für die Begleitung dieser Arbeit bedanken. Für die fachliche Unterstützung seitens der Firma AVL möchte ich mich besonders bei Herrn Gerhard Storfer bedanken.

Außerdem möchte ich allen Freunden und Bekannten, die mir eine mentale Stütze waren, meinen Dank zukommen lassen. Weiteres danke ich Christiane Wittmer und Karina Pinter für das Korrekturlesen der Diplomarbeit.

Abschließend möchte ich meiner Familie einen großen Dank aussprechen, die mich in meinem Studium finanziell und in vielen anderen Belangen unterstützt hat.

Inhaltsverzeichnis

1 Einleitung

Ein Zitat des Trend- und Zukunftsforschers John Naisbitt lautet: „We are drowning in information, but starving for knowledge“. Es beschreibt sehr gut den Umstand, dass wir mit stetig wachsenden Datenmengen konfrontiert sind und gleichzeitig ein Bestreben haben, Wissen aus den Daten zu erlangen. Das Interesse am potentiell nützlichen Wissen in Daten ist häufig auch wirtschaftlicher Natur. Diese Arbeit beschäftigt sich mit dem Bestreben nach Wissensgewinnung in einem betrieblichen Umfeld.

1.1 Aufgabenstellung

AVL CRETA™ ist eine Software der AVL List GmbH zur Verwaltung von Kalibrierdaten. Sie speichert diese zentral in einem Datenbanksystem. Seitens des Unternehmens existiert die Vermutung, dass in diesen Daten ungenutzte Informationen vorhanden seien, die einen wirtschaftlichen Nutzen haben.

Die Aufgabenstellung dieser Arbeit ist es, dieser Vermutung nachzugehen und anhand von Data Mining Methoden verwertbares Wissen aus den Daten zu extrahieren. Letztendlich soll ein Prototyp entwickelt werden, welcher einen Anwendungsfall implementiert, der für das Produkt einen Mehrwert darstellt.

1.2 Struktur der Arbeit

Die Arbeit beginnt mit einer knappen Beschreibung der Software AVL CRETA™, um dem Leser ein nötiges Grundverständnis zu vermitteln. Es wird auf die wichtigsten Kernfunktionen der Software eingegangen und es werden die Entwicklung der Datenmengen in den letzten Jahren aufgezeigt.

Im darauf folgenden Kapitel wird eine Einführung in die wichtigsten theoretischen Grundbegriffe gegeben. Neben der Einführung in den Wissensbegriff werden der Begriff Data Mining definiert sowie die Grundanforderungen des Data Mining erläutert. Außerdem wird ein Überblick darüber geboten, welche Data Mining Methoden existieren und wie sich diese einteilen lassen.

Anschließend wird auf unterschiedliche Data Mining Prozesse eingegangen und deren Gemeinsamkeiten sowie Unterschiede werden einander gegenübergestellt. Aufgrund der Tatsache, dass der CRISP-DM Prozess als Vorgehensmodell für die vorhandene

Aufgabenstellung gewählt wurde, werden die einzelnen Schritte dieses Prozessmodells skizziert.

Das nachfolgende Kapitel liefert das theoretische Hintergrundwissen zu den Algorithmen, die im Prototyp verwendet werden. Es werden zuerst zwei Algorithmen zur Assoziationsanalyse beschrieben und anschließend zwei Methoden zur Ausreißererkennung in Transaktionsdaten vorgestellt mit welchen die Ergebnisse dieser beiden Algorithmen weiterverarbeitet werden können. Außerdem wird auf die Regelgenerierung sowie auf Interessantheitsmaße von Regeln eingegangen.

Das anknüpfende Kapitel enthält einzelne Prozessschritte des CRISP-DM Prozesses im Zusammenhang mit der praktischen Umsetzung der Aufgabenstellung. Eingangs werden die Anforderungen aus betrieblicher Sicht dargestellt, dann folgt eine Beschreibung der zu analysierenden Daten. Aus den daraus gewonnenen Erkenntnissen wird eine Data Mining Methode bestimmt, die den Kern des Prototyps bilden soll.

Bevor die Data Mining Methode Anwendung finden kann, werden die Ausgangsdaten noch einer Aufbereitung unterzogen. Nach dieser Datenaufbereitung und der Methodenwahl erfolgt eine Beschreibung der im Model angewandten Algorithmen. Die Ergebnisse werden anschließend diskutiert und bewertet.

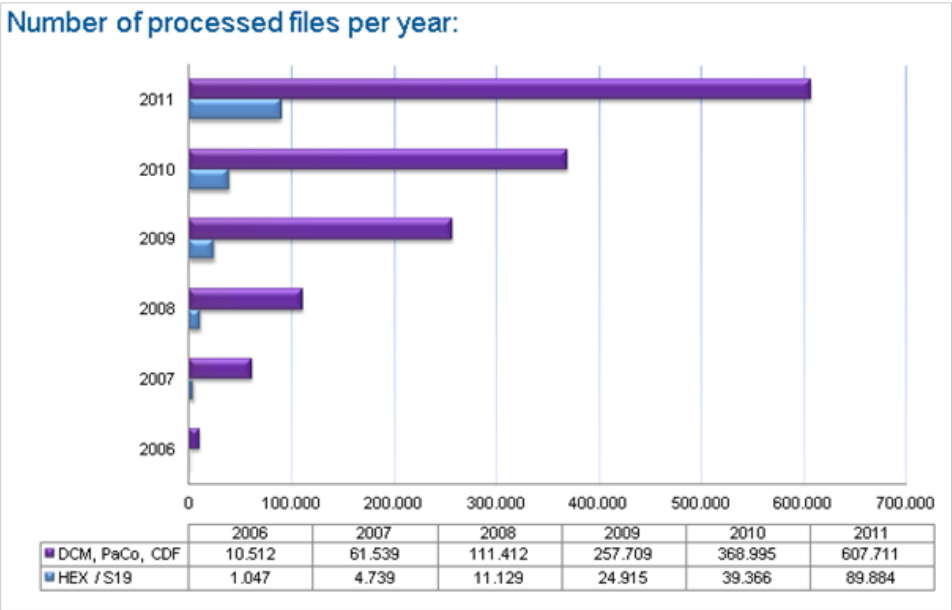
Um diese Ergebnisse noch zu verbessern, werden Ansätze zur Weiterverarbeitung aufgeführt und im Prototyp erprobt. Letztendlich erfolgen eine Evaluierung des Prototyps mit realen Daten und die Bewertung der Endergebnisse. Eine knappe Beschreibung der Benutzeroberfläche des Prototyps schließt das Kapitel ab.

Im letzten Abschnitt der vorliegenden Arbeit werden – in Form einer Conclusio – die Stärken und Schwächen des Systems diskutiert und es wird ein Ausblick in Bezug auf mögliche Weiterentwicklungen gegeben.

2 Über AVL Creta™

In einem Automobil ist heutzutage eine Reihe von Steuergeräten verbaut, welche diverse Eigenschaften eines Fahrzeuges beeinflussen. Durch die Kalibrierung der Steuergeräte werden beispielsweise Kernfunktionen wie Leistung, Emission, Verbrauch und Fahrverhalten konfiguriert. Die Anzahl der Steuergeräte sowie die Menge der konfigurierbaren Parameter nehmen jährlich zu. In Abbildung 1 ist dieser starke Zuwachs der verwalteten Kalibrierdaten eines Automobilherstellers in den Jahren 2006 bis 2011 dargestellt. Vor einem Jahrzehnt war die Anzahl dieser Parameter noch überschaubar und eine Verwaltung ohne ein eigenes Tool machbar. Dies ist heute undenkbar. Mittlerweile arbeiten große Projektteams mit Verantwortlichkeiten für einzelne Teilbereiche an solchen Kalibrierprojekten. Die Software AVL CRETA™[1] entstand aus diesem Grund und nimmt sich der diversen Problemstellungen während eines Kalibrierprozesses an. Die Kernfunktionen der Software sind laut (AVL CRETA™ Calibration Data Management, n.d.):

- Die zentrale Speicherung aller Kalibrierdaten und anderer projektrelevanter Daten
- Konfliktfreies Zusammenführen dieser Parameter
- Nachvollziehbarkeit des Kalibrierfortschrittes
- Regeln von Zugriffsrechten und Verantwortlichkeiten
- Erstellen einer Historie um alle Änderungen nachzuvollziehen



	2006	2007	2008	2009	2010	2011
DCM, PaCo, CDF	10.512	61.539	111.412	257.709	368.995	607.711
HEX / S19	1.047	4.739	11.129	24.915	39.366	89.884

Abbildung 1: Zunahme der Kalibrierdaten in der Motorenentwicklung (Storfer, 2012).

[1] https://www.avl.com/creta

Die grundlegende Arbeitsweise dieser Anwendung kann mit einem Versionsverwaltungssystem verglichen werden: Es gibt eine Historie der Änderungen, alte Datenstände können wieder hergestellt werden, der Zugriff von mehreren Personen auf die Daten ist geregelt und es gibt verschiedene unterschiedliche Zweige der Entwicklung.

3 Theoretische Einführung

Als Einstieg in dieses Themengebiet wird auf den Wissensbegriff, die Begriffsdefinition des Data Mining und auf die bedeutendsten Aufgabentypen des Data Mining eingegangen. Zuletzt werden noch die theoretischen Grundlagen für eine gängige Vorgehensweise bei Data Mining Projekten, dem CRISP-DM Prozess, skizziert.

3.1 Wissensbegriff

Das Ziel des Data Mining ist es, Wissen aus Daten zu gewinnen. In Abbildung 2 ist die Wissenspyramide illustriert. Sie soll den allgemeinen Weg zur Wissensgewinnung, die Abgrenzung der Begriffe und die Übergänge der Phasen veranschaulichen.

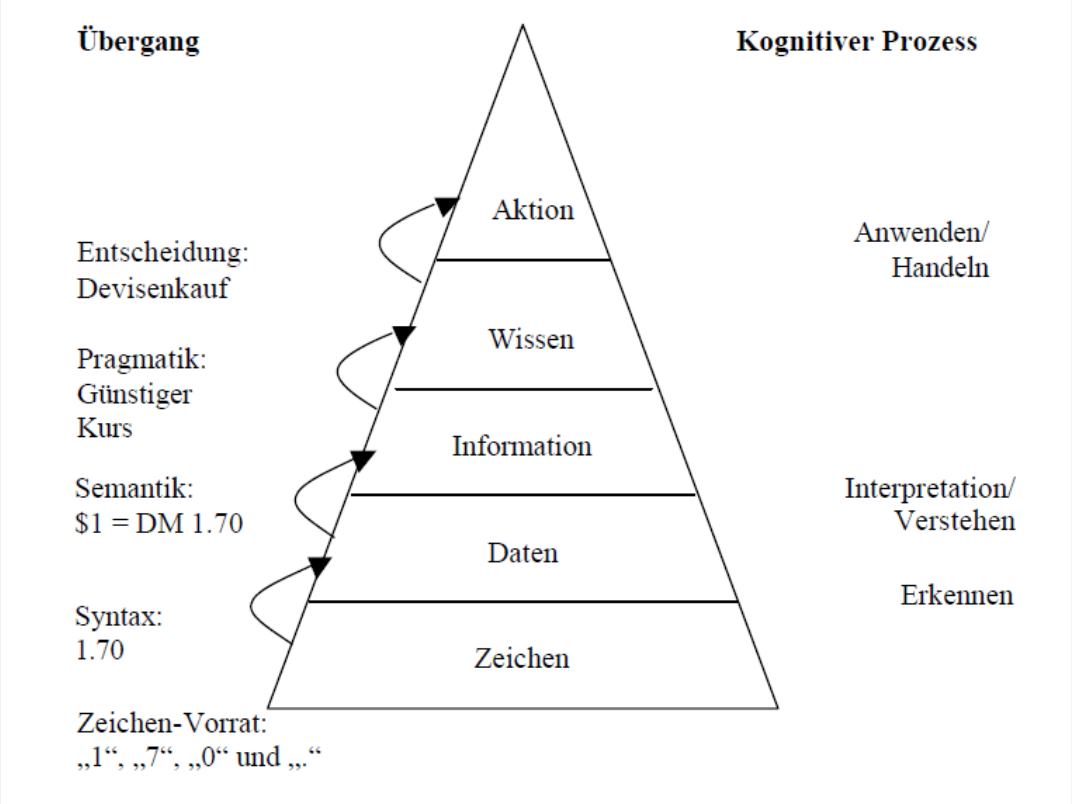


Abbildung 2: Wissenspyramide nach (Fuchs-Kittowski, 2002, S. 27).

Nach (Cleve, 2011, S. 11) lassen sich die Begriffe wie folgt definieren:

- Die *Zeichen* stellen die kleinste Darstellungseinheit dar
- *Daten* sind die Ansammlung von Zeichen mit der entsprechenden Syntax. Es wird zwischen unstrukturierten Daten (Bild, Text), semistrukturierten Daten (Webseiten) und strukturierten Daten (Datenbanken) unterschieden

- Von *Information* wird gesprochen, wenn Daten mit einer Bedeutung verbunden werden
- *Wissen* ist eine Information in Zusammenhang mit der Kompetenz, diese auch benutzen zu können

3.2 Begriff des Data Mining

Der Data Mining Prozess wird in der Literatur häufig als Synonym für *Knowledge Discovery in Databases* (kurz KDD) benutzt. In dieser Arbeit werden die beiden Begriffe auch sinngleich verwendet. Die Bezeichnung Data Mining wird als ein Teilschritt des KDD Prozesses verstanden, der sich unterschiedlicher Analysealgorithmen bedient. Die beiden Begriffe Data Mining und KDD sind nach (Fayyad, Piatetsky-Shapiro, & Smyth, 1996, S. 40ff) folgendermaßen definiert:

> *"KDD is the **nontrivial process** of identifying **valid**, **novel**, **potentially useful**, and **ultimately understandable** patterns in data."*

> *"Data mining is a step in the KDD process that consists of applying **data analysis** and **discovery algorithms** that, under **acceptable computational efficiency limitations**, produce **a particular enumeration of patterns** (or models) over the data."*

3.3 Methoden des Data Mining

Eine Herausforderung in der Datenanalyse ist es, aus der Fülle von Data Mining Methoden die passende für die jeweilige Problemstellung zu finden. Um dies zu erleichtern gibt es in der Praxis eine Einteilung der Data Mining Problemstellungen in Prognoseprobleme und Beschreibungsprobleme. Durch diese Einteilung können bestimmte Methoden bereits aufgrund der Datenbasis oder des gewünschten Ergebnisses ausgeschlossen werden. Prognoseprobleme versuchen anhand von bekannten Merkmalen eines Informationsobjektes eine Aussage über unbekannte oder künftige Merkmalswerte herzuleiten. Beschreibungsprobleme fokussieren darauf, Informationen und Muster in den Daten aufzuspüren, welche für den Menschen nachvollziehbar, interpretierbar und handlungsrelevant sind. Es sei noch erwähnt, dass die Grenzen zwischen diesen unterschiedlichen Problemstellungen nicht strikt sind. Es kommt durchaus vor, dass Prognosemodelle auch einen beschreibenden Charakter haben und umgekehrt, dass aus beschreibenden Modellen Prognosen abgeleitet werden. Trotz dieser zum Teil unscharfen Abgrenzung kann diese Einteilung als Hilfestellung zum Auffinden der richtigen Methode

gesehen werden. (Fayyad, Piatetsky-Shapiro, & Smyth, 1996, S. 44) (Hippner, Küsters, Meyer, & Wilde, 2001, S. 63)

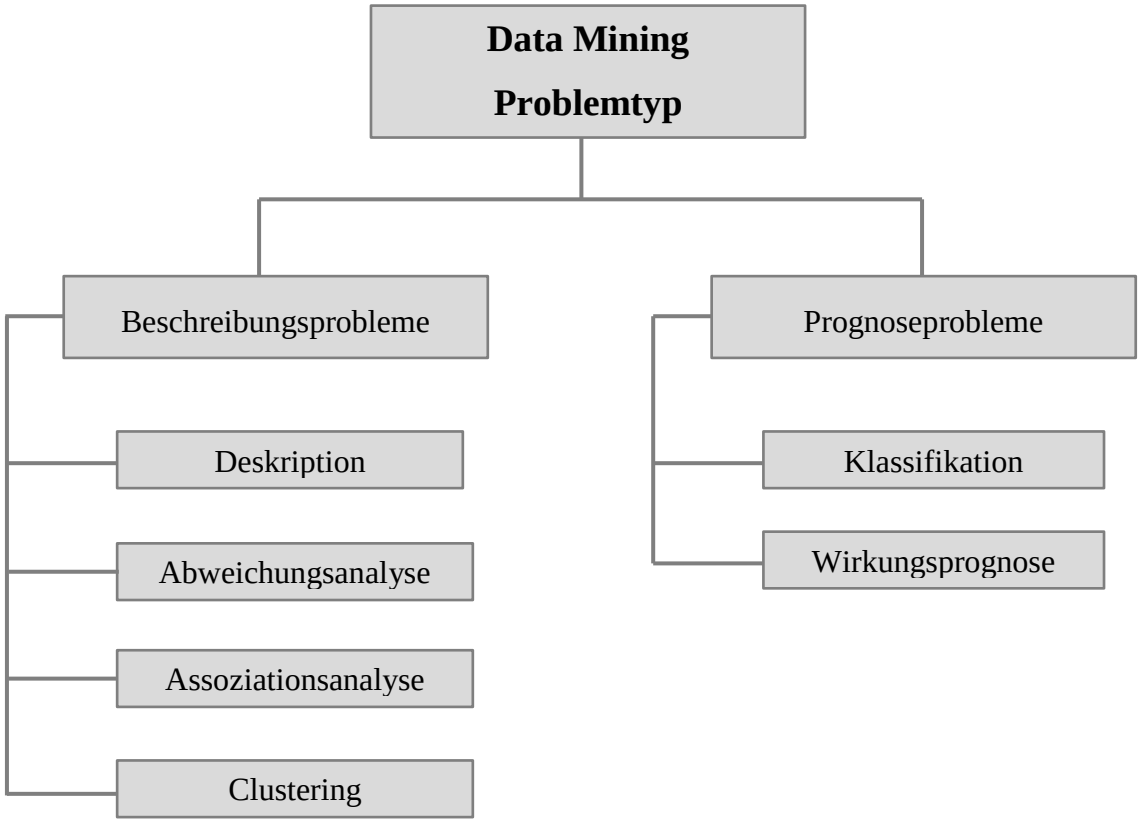


Abbildung 3: Problemtypen des Data Mining (Hippner, Küsters, Meyer, & Wilde, 2001, S. 64).

Wie in Abbildung 3 dargestellt gibt es innerhalb dieser beiden Problemtypen eine Unterteilung in weitere Kategorien. Diese Kategorien werden nachfolgend kurz beschrieben, um die spätere Auswahl der Methode für den Prototyp nachvollziehbar zu machen.

3.3.1 Deskription

Die Technik der Deskription wird nicht als eine reine Data Mining Methode angesehen, da sie an der Grenze zwischen der Datenaufbereitung und Exploration steht. Jedoch können durch die Deskription Muster entdeckt werden, weswegen sich inhaltlich zweifelsohne eine Nähe zum Begriff des Data Mining ergibt. (Gabriel, Gluchowski, & Pastwa, 2009, S. 136)

Bei diesem Verfahren liegt der Fokus nicht in den ursprünglichen, meist sehr umfang- und detailreichen Daten, sondern in der Beschreibung aufschlussreicher Strukturen mit deskriptiven statistischen Methoden, Visualisierungsmethoden oder OLAP. (Hippner, Küsters, Meyer, & Wilde, 2001, S. 64)

OLAP steht für *Online Analytical Processing*, ist eine typische Generalisierungsanwendung und wird auch als eine Art manuelles Data Mining bezeichnet. Die Daten werden in der Regel aus der operativen Datenbank in einen sogenannten OLAP Würfel überführt, aus welchem mit einfachen Abfragen diverse Analysen durchgeführt werden können. Diese Methode findet oftmals im Management Einsatz. Ein beispielhafter Anwendungsfall wäre die Analyse der Leistung verschiedener Filialen indem der Benutzer mit einigen Basisoperationen diese Informationen ausliest. (Ester & Sander, 2000, S. 189)

3.3.2 Abweichungsanalyse

Die Aufgabe der Abweichungsanalyse ist es, Objekte aufzufinden, die von einer Norm oder einem Erwartungswert abweichen. Mit dem Verfahren der Abweichungsanalyse können auffällige Entwicklungen oder Ausreißer in einer Datenbasis detektiert werden. Die Abweichungsanalyse steht zum Teil ebenfalls an der Grenze zur Datenaufbereitung, weshalb auch ihre Zuordnung zu den Data Mining Verfahren nicht unumstritten ist. Die Abweichungsanalyse kann jedoch auch zur Erfassung zuvor unbekannter Datenmuster verwendet werden, wodurch sie den Anforderungen eines Data Mining Verfahrens teilweise gerecht wird. (Gabriel, Gluchowski, & Pastwa, 2009, S. 136ff)

3.3.3 Assoziationsanalyse

Assoziationsanalysen werden für die Beschreibung von Abhängigkeiten zwischen Objekten einer Datenbasis herangezogen. Ein klassisches Beispiel der Assoziationsanalyse ist die sogenannte Warenkorbanalyse. Die nötigen Lerndaten für Assoziationsanalysen sind eine Sammlung an Warenkörben, die Kundentransaktionen darstellen, welche beispielsweise eine Menge an zusammengekauften Artikeln sein könnten. Ein allgemeines Beispiel für solche Transaktionsdaten ist die Sammlung der Daten, welche durch die Barcode-Scanner an Supermarktkassen erhoben werden können. Eine einzelne Transaktion wäre die Menge an Artikeln, die ein Kunde bei einem Einkauf an der Kasse bezahlt hat. (Ester & Sander, 2000, S. 159)

Assoziationsregeln drücken die Zusammenhänge innerhalb von Transaktionen aus, die in der gesamten Datenmenge häufig vorkommen. Ein einfaches Beispiel für eine Assoziationsregel wäre: Liegen Eier und Mehl in einem Warenkorb, dann wird häufig auch noch Butter im Warenkorb vorkommen. (Ester & Sander, 2000, S. 159)

Assoziationsregeln finden häufig Anwendung in den klassischen Warenkorbanalysen, beschränken sich jedoch nicht nur auf diese. Wie die zuvor genannten

Beschreibungsproblemtypen steht auch die Assoziationsanalyse an der Grenze zur Datenaufbereitung. (Hippner, Küsters, Meyer, & Wilde, 2001, S. 65)

3.3.4 Clustering

Die Aufgabe der Clusteranalyse ist es, eine gegebene Menge in verschiedene homogene Teilmengen zu zerlegen. Die Elemente innerhalb eines Custers sollen sich möglichst ähnlich sein. Elemente unterschiedlicher Cluster sollten hingegen möglichst unähnlich sein. Ein typisches Beispiel für Clustering wäre das Auffinden möglichst homogener Kundengruppen für eine gezielte Angebotsgestaltung. (Cleve, 2011, S. 18ff)

3.3.5 Klassifikation

Die Aufgabe der Klassifikation ist die Einteilung von Informationsobjekten (z.B. Kunden) in bestimmte Klassen, die über Merkmale definiert sind (z.B. schlechte / normale / gute Kreditwürdigkeit). Die Bestimmung der Kreditwürdigkeit von Kunden ist für diese Problemstellung ein gängiges Beispiel. Anhand einer Trainingsmenge von zuvor bekannten Instanzen, welche Attributwerte beinhalten (z.B. Alter, Einkommen etc.), wird ein Modell generiert. Dieses Modell wird gegen Testdaten validiert und gegebenenfalls noch angepasst. In den Lerndaten sind sowohl die Attributwerte als auch die resultierende Klasse bekannt. Man spricht daher auch von überwachtem Lernen. Ist das gelernte Modell gut genug, kann dieses neue Instanzen anhand der Attributwerte in die dementsprechenden Klassen einteilen. (Cleve, 2011, S. 17ff) (Hippner, Küsters, Meyer, & Wilde, 2001, S. 65)

3.3.6 Wirkungsprognose

Analog zur Klassifikation ist es das Ziel der Wirkungsprognose, einen Zielwert vorauszusagen. Bei der Wirkungsprognose wird jedoch ein Zielwert mit numerischer Ausprägung bestimmt. Das Trainieren des Modelles erfolgt ebenfalls über Instanzen mit bekannten Attributen und Zielwerten sowie über eine anschließende Validierung gegen Testdaten mit einer eventuell nachfolgenden Korrektur des Modells. Ein Beispiel für den Einsatz von Wirkungsprognosen wäre die Vorhersage des geschätzten Auftragsvolumens eines Kunden für die nächste Saison auf Basis seiner bisherigen Verkaufshistorie. Typische Methoden der Wirkungsanalyse sind Regressionsanalysen. (Hippner, Küsters, Meyer, & Wilde, 2001, S. 65ff)

3.4 Prozess des Data Mining

Um einen Leitfaden für die Vorgehensweise zum Lösen einer Data Mining Problemstellungen zu finden wurde versucht, sich an vorhandenen Data Mining Prozessmodellen orientieren. Die drei gängigsten Modelle sind der KDD-, SEMMA- und CRISP-DM Prozess. (Azevedo & Santos, 2008, S. 182)

Diese drei Modelle sind sich im Grundaufbau sehr ähnlich. In Tabelle 1 sind die einzelnen Schritte der Prozesse gegenübergestellt. Jeder dieser drei Prozesse würde die Anforderung erfüllen eine Data Mining Problemstellung zu lösen, jedoch wurde für die Problemstellung dieser Arbeit auf den CRISP-DM zurückgegriffen.

Diese Entscheidung ist damit begründet, dass der CRISP-DM Prozess einerseits am besten dokumentiert ist und andererseits auch das Business Understanding berücksichtigt. Außerdem spricht für dieses Verfahren, dass es aus einer praktischen Anwendung heraus entstanden ist und eine gute Grundlage für Data Mining Problemstellungen in einem betrieblichen Umfeld bietet.

KDD	SEMMA	CRISP
Pre KDD	----------	Business Understanding
Selection	Sample	Data Understanding
Pre Processing	Explore	
Transformation	Modify	Data Preperation
Data Mining	Model	Modeling
Interpretation/Evaluation	Assesment	Evaluation
Post KDD	----------	Deployment

Tabelle 1: Vergleich KDD, SEMMA und CRISP nach (Azevedo & Santos, 2008).

Im folgenden Kapitel werden die einzelnen Phasen des CRISP-DM Prozesses kurz beschrieben. Für einen Vergleich dieser drei Prozessmodelle wird auf (Azevedo & Santos, 2008) verwiesen.

3.5 Der CRISP-DM Prozess

CRISP-DM ist die Abkürzung für CRoss-Industry Standard Process for Data Mining. Dieser Prozess entstand aus einem Konsortium der drei Unternehmen DaimlerChrysler, SPSS und NCR und wurde auch später von der Europäischen Union über das ESPRIT[2]-Programm finanziell unterstützt. (Chapman, et al., 2000, S. 1)

Das CRISP-DM Prozessmodell beschreibt eine allgemeine Herangehensweise an Data Mining Problemstellungen und will eine Hilfestellung bzw. einen allgemeinen Leitfaden für Data Mining Projekte geben. Das Prozessmodell gibt einen generell gültigen Überblick über den Lebenszyklus eines Data Mining Projektes. (Chapman, et al., 2000, S. 10)

Die Abbildung 4 zeigt des CRISP-DM Prozess Modell, welches die vorhandenen Phasen eines Data Mining Projektes mit dessen Beziehungen zueinander skizziert. Anschließend wird das Modell kurz erläutert und eine knappe Zusammenfassung der einzelnen Phasen aufgelistet.

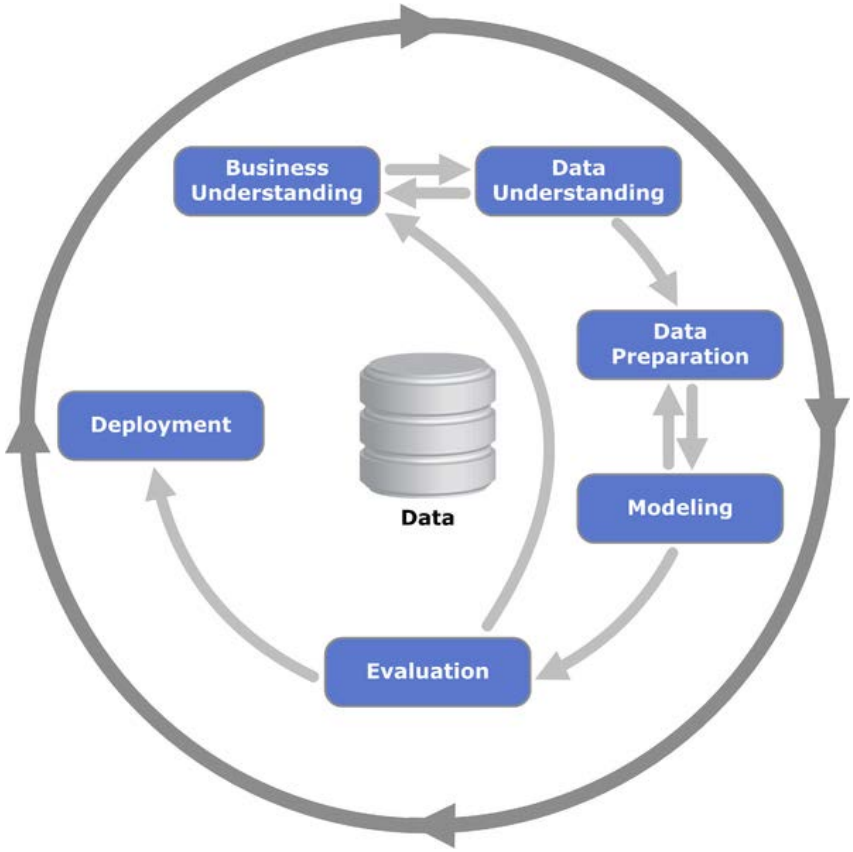


Abbildung 4: CRISP-DM Prozess Modell nach (Chapman, et al., 2000, S. 10).

[2] European Strategic Program on Research in Information Technology: http://cordis.europa.eu/esprit/

Ein Data Mining Projekt wird als zyklischer sechsstufiger Prozess beschrieben. Die Abfolge der einzelnen Phasen ist nicht absolut statisch und somit ist ein Vor- und Zurückbewegen möglich bzw. sogar erwünscht. Die Pfeile beschreiben die wichtigsten und häufigsten Abhängigkeiten zwischen den Phasen. Der äußere Kreis beschreibt die zyklische Natur des Data Mining, da es nicht einfach mit dem Schritt des Deployment endet. Es können sich nach einer ersten Iteration gezieltere oder sogar neue Geschäftsideen aus dem Data Mining Projekt entwickeln. (Chapman, et al., 2000, S. 10)

3.5.1 Business Understanding

(Chapman, et al., 2000, S. 30ff)

- Festlegen von Unternehmenszielen: Es werden die Ziele des Data Mining Projektes aus Unternehmenssicht definiert und Kriterien festgelegt, an welchen die erfolgreiche Umsetzung des Projektes aus betrieblicher Sicht bewertet werden kann. Diese Zieldefinition soll verhindern, dass in einem Data Mining Projekt ein Aufwand betrieben wird, der zwar technisch richtige, aus unternehmerischer Sicht jedoch irrelevante Ergebnisse liefert.
- Bewerten der Ausgangssituation: In diesem Schritt erfolgt das Erfassen von vorhandenen Ressourcen, möglichen Einschränkungen, Annahmen und Faktoren, die einen Einfluss auf das Projektziel haben können. Es erfolgt die Feststellung von Schlüsselpersonen wie Anwender, Geschäfts-, Daten- und Data Mining Experten sowie eine Identifikation der Hardware- und Softwareressourcen und der Datenquellen. Mögliche Risiken und organisatorische Hindernisse, wie beispielsweise Zugriffsrechte und Datensicherheit müssen eingeplant werden.
- Festlegen von Data Mining Zielen: Im Gegensatz zu den unternehmerischen Zielen erfolgt an dieser Stelle eine technischere Formulierung der Ziele. Auch die erwarteten Ergebnisse werden aus technischer Sicht formuliert, um die Zielerreichung messbar zu machen.
- Erstellen eines Projektplanes: Im Projektplan sollen die Phasen des Projektes aufgelistet und ihre Abhängigkeiten, Dauer, benötigte Ressourcen sowie mögliche Risiken detailliert behandelt werden. Unter anderem soll der Projektplan kritische Abschnitte, Entscheidungspunkte und Zeitpunkte für Reviews festlegen. Im Projektplan sind auch mögliche Tools und Techniken beschrieben und anhand welcher Kriterien diese ausgewählt wurden.

3.5.2 Data Understanding

(Chapman, et al., 2000, S. 37ff)

- Sammeln der Ausgangsdaten: In diesem Schritt soll der erste Zugriff auf die Daten erfolgen. Es wird festgestellt, wie die Daten geladen werden, woraus sich schon erste Schritte zur Datenaufbereitung ergeben können.
- Beschreiben der Daten: Es wird ein Überblick über das Format der Daten, die Datenmenge, Attribute, Attributtypen gegeben und festgestellt ob die Daten überhaupt die erwarteten Anforderungen erfüllen können.
- Untersuchen der Daten: In diesem Schritt werden erste Versuche mit den Daten durchgeführt und daraus Hypothesen abgeleitet, welche in Data Mining Ziele münden können. Die Hypothesen werden mit einfachen Analysen verifiziert und Data Mining Ziele werden verfeinert bzw. neu definiert.
- Verifizieren der Datenqualität: In diesem Abschnitt wird die Qualität der Daten beleuchtet. Es gibt Fragestellungen wie: Sind die Daten vollständig? Gibt es Fehler in den Daten? Gibt es fehlende Werte in den Daten? Welche Bedeutung haben fehlende Werte? Welche Bedeutung haben die Attribute? Gibt es fehlende Attribute?

3.5.3 Data Preperation

(Chapman, et al., 2000, S. 42ff)

- Selektion der Daten: In diesem Abschnitt wird beschlossen, welche Daten verwendet werden. Abhängig von der zuvor analysierten Datenqualität werden Datensätze inkludiert oder exkludiert. Außerdem wird eine Entscheidung getroffen, welche Attribute mehr oder weniger bedeutend sind, woraus auch eine Gewichtung der Attribute abgeleitet werden kann.
- Datenbereinigung: Die Datenqualität wird weiter erhöht und es erfolgen erste Anpassungen an die verwendete Analysetechnik. Fehlende Werte werden mit Default Werten ersetzt oder geschätzt. Es wird entschieden, wie mit Datenrauschen umgegangen wird (korrigieren, entfernen, ignorieren).
- Erstellen von Daten: In diesem Schritt können beispielsweise aus vorhandenen Attributen neue Attribute abgeleitet werden (z.B. Fläche = Länge x Breite). Es fällt die Entscheidung, ob Attribute normalisiert oder Werte transformiert werden müssen. (z.B. nominal zu numerisch oder es können aus vorhanden Datensätzen völlig neue Datensätze generiert werden)

- Integrieren von Daten: Hier können beispielsweise Informationen von mehreren Tabellen kombiniert werden, um wieder neue Datensätze zu erhalten oder man kann zusätzliche Informationen einfließen lassen, welche eventuell auch in einem nicht-elektronischen Format vorliegen.
- Formatieren der Daten: Die Daten werden nur formatiert ohne ihre Bedeutung zu ändern. Ein Beispiel dafür wäre eine Sortierung der Attribute oder die Verschiebung des vorherzusagenden Klassenattributs an die letzte Stelle. Dasselbe gilt auch für die Datensätze. Wenn es vom Modeling Tool verlangt wird, dann werden auch diese sortiert.

3.5.4 Modeling

(Chapman, et al., 2000, S. 49ff)

- Wählen der Modelltechnik: Es erfolgt die Auswahl der entsprechenden Modelltechnik (Klassifikation, Clustering, Assoziationsanalyse etc.). Die Wahl der Technik muss begründet werden. Häufig können auch aufgrund der Grunddaten (Qualität, Format) bestimmte Techniken ausgeschlossen werden.
- Erstellen eines Testdesign: Das Testdesign soll das Trainieren, Testen und Evaluieren des Modells planen. Eine wichtige Komponente ist die Einteilung der Daten in Trainings- und Testdaten sowie das Festlegen von Maßstäben um Qualitätsaussagen über die Modelle machen zu können (Fehlerraten, mittlerer quadratischer Fehler, Kappa Statistik etc.)
- Erstellen eines Modells: In dieser Phase werden Modelle erstellt. Abhängig vom Modeling Tool kann das Modell über diverse Parameter eingestellt werden. Hier sollen die Einstellungen dokumentiert und begründet werden. Am Ende soll das Ergebnis beschrieben werden.
- Bewerten des Modells: Die Ergebnisse werden Abhängig von den zuvor definierten Data Mining Erfolgskriterien bewertet. Hier kann die Genauigkeit eines Modells festgestellt bzw. können die Resultate überprüft werden (Trainingsdaten vs. Testdaten, Cross-validation). Es empfiehlt sich, auch Meinungen von Domänenexperten einzuholen um Modelle zu bewerten. Modelle werden hinsichtlich der gesetzten Data Mining Ziele überprüft und daraus lässt sich ein Ranking für unterschiedliche Modelle erstellen.

3.5.5 Evaluation

(Chapman, et al., 2000, S. 51ff)

- Evaluieren der Ergebnisse: In diesem Schritt wird bewertet inwieweit das Modell die anfangs gesetzten Unternehmensziele erreichen konnte. Eine weitere Möglichkeit der Evaluierung wäre es, die Modelle in einer realen Anwendung zu testen sofern dies zeitlich und budgetär möglich ist. Es werden auch andere Erkenntnisse festgehalten, vorausgesetzt sie sind von Bedeutung für die vorhandenen sowie für mögliche neue Geschäftsziele. Nach der Evaluierung besteht die Möglichkeit, dass die definierten Geschäftsziele entsprechend den gewonnenen Erkenntnissen angepasst werden müssen.
- Review des Prozesses: Zu diesem Zeitpunkt hat das Modell die Anforderungen ausreichend erfüllt. Es empfiehlt sich, nun noch einmal jede Stufe des Data Mining Prozesses zu beleuchten um sicherzustellen, dass nicht irgendein bedeutender Faktor übersehen wurde. Der Review des gesamten Prozesses hat eine qualitätssichernde Aufgabe und kann wichtige Informationen für zukünftige weitere Iterationen liefern. Fragestellungen an die diversen Phasen sind: War diese notwendig? Wurde sie optimal ausgeführt? In welcher Art und Weise könnte sie verbessert werden?
- Festlegen der nächsten Schritte: Die zentrale Fragestellungen an diesem Punkt sind, ob bereits in die Phase des Deployments übergegangen werden kann, ob weitere Iterationen gestartet werden oder ob überhaupt ein neues Data Mining Projekt entsteht.

3.5.6 Deployment

(Chapman, et al., 2000, S. 54ff)

- Planen des Einsatzes: Nachdem die Ergebnisse evaluiert wurden muss eine Strategie zurechtgelegt werden wie diese nun unternehmerisch zum Einsatz kommen.
- Planen von Kontrolle und Wartung: Bevor es noch zum realen Einsatz des Systems kommt, sollte ein Plan für Kontrolle und Wartung entworfen werden. Fehlt dieser Plan besteht die Möglichkeit, dass Modelle nicht zeitgerecht aktualisiert werden und möglicherweise mit falschen Ergebnissen gearbeitet wird.
- Abschlussreport erstellen: Am Ende des Projektes wird noch ein Abschlussreport erstellt, der eine Zusammenfassung des Projektes mit den gewonnen Erkenntnissen sowie eine Endpräsentation der Data Mining Ergebnisse beinhaltet.

- Review des Projektes: Zuletzt wird noch ein Review Report erstellt, welcher kurz zusammenfasst was gut bzw. was schlecht gelaufen ist und welche Erfahrungen während des Projektes gewonnen wurden, um diese auch für zukünftige Projekte nutzen zu können.

4 Algorithmische Grundlagen

Im 5ten Kapitel, welches sich mit der praktischen Umsetzung beschäftigt und am CRISP-DM Prozess angelehnt ist, wird befunden, dass die Assoziationsanalyse eine geeignete Methode für die gegebene Problemstellung darstellt. Daher werden in diesem Abschnitt die wichtigsten theoretischen Grundlagen der Assoziationsanalyse und notwendige Definitionen erläutert.

Anschließend werden die Arbeitsweise des Apriori Algorithmus und jene des FP-Growth Algorithmus beschrieben und miteinander verglichen. Die Auswahl dieser beiden Algorithmen ist einerseits damit begründet, dass der Apriori ein Standardalgorithmus[3] für Assoziationsanalysen ist. Andererseits ist der FP-growth[4] ist ein jüngerer Algorithmus, welcher nach Aussagen seiner Entwickler leistungsfähiger als der Apriori sein soll. Anschließend wird der Algorithmus zur Regelgenerierung beschrieben und es werden diverse Interessantheitsmaße aufgezeigt, um möglichst starke Regeln zu extrahieren.

Da die Assoziationsanalyse meist eine für den Menschen kaum überschaubare Ergebnismenge liefert, werden noch zwei Methoden vorgestellt, die diese Ergebnisse weiterverarbeiten. Dazu werden zwei Algorithmen skizziert, welche nach möglichen Ausreißern in Transaktionsdaten suchen.

4.1 Assoziationsanalyse

4.1.1 Begriffsdefinitionen

Die Bildung von Assoziationsregeln lässt sich in folgende zwei Schritte unterteilen:

1. Finden der häufigen *Itemmengen*
2. Generieren der Regeln aus den gefundenen *Itemmengen*

Vor der Erklärung der Algorithmen werden einleitend die wichtigsten Grundbegriffe aufgelistet um ein formales Verständnis für Assoziationsregeln zu vermitteln und die anschließend beschriebenen Algorithmen besser verstehen zu können.

Die folgenden Definitionen wurden aus (Ester & Sander, 2000, S. 160) und (Narita & Kitagawa, 2008) entnommen:

[3] Laut (Witten, Frank, & Hall, 2011, p. 376) zählt der Apriori Algorithmus zu den Top 10 Data Mining Algorithmen.
[4] (Han, Pei, & Yin, 2000) über die Eigenschaften des FP-Growth Algorithmus.

- **I** $=\{i_1,...i_m\}$ ist die Menge von unterschiedlichen *Items*, die in einer Transaktionsdatenbank existieren. Ein *Itemset* X ist eine Menge von *Items*, für welche gilt, dass $X \subseteq I$ ist
- **T** steht für eine Menge von Transaktionen, in der Regel ist damit die Transaktionsdatenbank gemeint
- **|T|** repräsentiert die Anzahl sämtlicher Transaktionen in der Transaktionsdatenbank
- **t** ist eine einzelne Transaktion aus T
- **k** repräsentiert die Länge eines *Itemsets*, ein *Itemset* der Länge k wird auch k-*Itemset* bezeichnet
- $\mathbf{L_k}$ ist ein *Large-* bzw. *Frequent-Itemsets* mit der Länge k, für dieses *Itemset* gilt, dass es einen definierten minimalen *Support* (*min_sup*) erfüllt
- **min_sup** ist ein in Prozent angegebener Schwellwert und beschreibt die mindestens geforderte relative Häufigkeit eines *Itemsets* in der Transaktionsdatenbank um ein *Large-Itemset* zu sein
- Der *Support* eines *Itemsets* X ist nach dem folgenden Ausdruck formuliert:

$$\boldsymbol{sup}(\boldsymbol{x}) = \frac{|\{\boldsymbol{t}|\boldsymbol{t} \in \mathbf{T} \ \wedge \mathbf{X} \subseteq \mathbf{t}\}|}{|\boldsymbol{T}|}$$

- $\mathbf{C_k}$ ist ein *Candidateset*. Dies ist eine Kombination von *Items*, welche ein mögliches *Large-Itemset* darstellt, dessen *Support* erst bestimmt werden muss
- Eine Assoziationsregel wird in der Form $A \rightarrow B$ geschrieben, wobei X und Y zwei *Itemsets* sind für die gilt $X \subseteq I$, $Y \subseteq I$ und $X \cap Y = \emptyset$
- Der *Support* einer Regel $A \rightarrow B$ ist der *Support* $\{A \cup B\}$ in T, dies ist die relative Häufigkeit des gemeinsamen Auftretens von A und B
- Die *Confidence* eine Assoziationsregel ist folgendermaßen definiert:

$$\boldsymbol{conf}(\boldsymbol{A} \rightarrow \boldsymbol{B}) = \frac{\mathbf{sup}(\mathbf{A} \cup \mathbf{B})}{\mathbf{sup}(\boldsymbol{A})}$$

- **min_conf** ist ein in Prozent angegebener Schwellwert für Assoziationsregeln, es werden nur Regeln übernommen die diesen Schwellwert erreichen

4.1.2 Der Apriori Algorithmus

Der Apriori Algorithmus wurde vom IBM Almaden Forschungszentrum entwickelt und ist ein gängiges Verfahren zur Bildung von Assoziationsregeln. Der Zweck dieses Algorithmus ist es, die Menge aller häufigen *Itemkombinationen* (engl. *itemsets*) zu erkennen. (Hippner, Küsters, Meyer, & Wilde, 2001, S. 430)

Die meisten weiteren Entwicklungen bauen auf diesen Algorithmus auf oder ziehen diesen aufgrund seiner bis heute noch akzeptablen Leistung in Vergleichsstudien ein. Im Vergleich zu vorangegangenen Verfahren wie SetM[5] und AIS[5] ist der Rechenaufwand vom Apriori Algorithmus wesentlich geringer. (Petersohn, 2005, S. 107)

Der Apriori Algorithmus arbeitet stufenweise und erstellt pro k-ter Iteration zuerst eine Menge C_k mit den jeweils möglichen *Large-Itemsets (*auch *Candidatesets* genannt*)* mit der Länge *k*. Jene *Itemsets*, welche den minimalen *Support* erfüllen, werden anschließend nach L_k, genannt *Large-* oder *Frequent-Itemsets*, übernommen. Bei der Erzeugung von *Large-Itemsets* nutzt man die Erkenntnis, dass alle Teilmengen eines *Large-Itemsets* ebenfalls *Large-Itemsets* sein müssen. (Petersohn, 2005, S. 108)

Es können daher die *Candidate k-Itemsets* aus den *Large^{k-1}-Itemsets* gebildet werden, was auch *Join-Step* genannt wird. Nach dem *Join-Step* erfolgt noch der *Prune-Step*, welcher sich ebenfalls das Wissen um die Monotonieeigenschaft von *Large-Itemsets* zu Nutzen macht. Die Monotonieeigenschaft besagt: Ist ein *Itemset* häufig, dann sind auch alle Teilmengen dieses *Itemsets* häufig. Diese Eigenschaft kann auch umgekehrt formuliert werden: Ist ein *Itemset* nicht häufig dann sind alle *Itemsets*, die eine Übermenge dieses *Itemsets* sind, ebenfalls nicht häufig. Aufgrund dieser Eigenschaft werden im *Prune-Step* jene *k-Candidateitems* aus dem *k-Candidateset* entfernt, welche eine (k-1) Teilmenge besitzen, die nicht häufig ist. (Petersohn, 2005, S. 108)

Im Wesentlichen lässt sich der Algorithmus laut (Petersohn, 2005, S. 108) in folgende drei Teilschritte untergliedern:

- 1. Schritt: Ausführen des *Join-Step* und *Prune-Step,* um das *k- Candidateset* C_k zu erstellen
- 2. Schritt: Errechnen des *Support* für alle *k- Candidatesets*
- 3. Schritt: Generieren der *k- Large-Itemsets* anhand jener *k- Candidatesets*, deren *Support* mindestens dem definierten *Minimumsupport* entspricht

Nachfolgend wird in Listing 1, in Form von Pseudocode, die grundlegende Funktionsweise des Algorithmus illustriert:

[5] Für Performancevergleiche zwischen AIS, SetM und dem Apriori Algorithmus sei hier auf (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 494) verwiesen.

Notation:

L_k: Set of large k-itemsets (those with minimum support).

C_k: Set of candidate k-itemsets (potentially large itemsets).

L_1 = {large 1-itemsets};
for (k=2; $L_{k-1} \neq \emptyset$; k++) **do begin**
    C_k = apriori-gen(L_{k-1}); *// new candidates*
    forall transactions t $\in$ D **do begin**
        C_t = subset(C_k,t); *// candidates contained in t*
        forall candidates c $\in C_t$ **do**
            c.count++;
    end
    L_k = {c $\in C_k$ | c.count $\geq$ minsup}
end
Answer = $\bigcup_k L_k$;

Listing 1: Apriori Algorithmus aus (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 489).

Listing 2 erläutert die *apriori-gen()* Funktion, in welcher über den *Join*- und *Prune-Step* die Bildung der k- *Candidatesets* erfolgt.

// Join Step
insert into C_k
select p.item$_1$, p.item$_2$, … p.item$_{k-1}$, q.item$_{k-1}$
from L_{k-1} p, L_{k-1} q
where p.item$_1$ = q.item$_1$, … p.item$_{k-2}$ = q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$

// Prune Step
forall itemsets c $\in C_k$ **do**
    forall (k-1)-subsets s of c **do**
    if (s $\notin$ L_{k-1}) **then**
        delete c from C_k;
    end
end

Listing 2: Apriori-gen Funktion nach (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 490).

Im Folgenden wird anhand eines Beispiels erläutert, wie der Apriori-Algorithmus die *Large-Itemsets* generiert:

Datenbank D

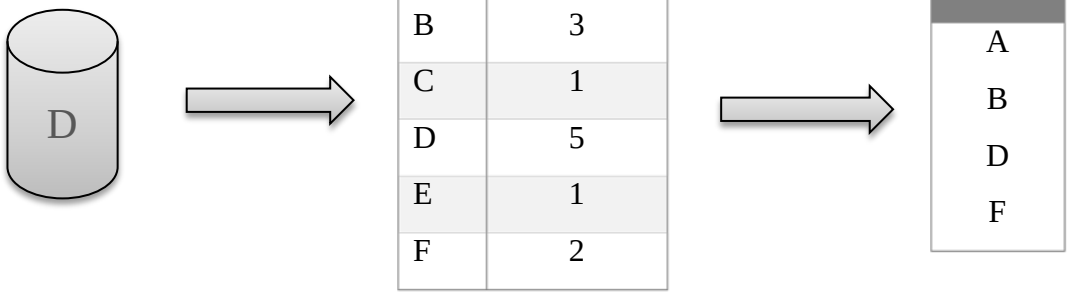
Transaktions-ID	Items
T1	A, C, D
T2	D
T3	A, B, D, E
T4	A, F
T5	A, B
T6	A, F
T7	A, D
T8	A, B, D

Gegeben sei eine Datenbank mit acht Transaktionen, welche die *Items* I = {A, B, C, D, E, F} beinhaltet. Die einzelnen Transaktionen haben eine eindeutige ID (T1…T8). Die *Items* sind in jeder Transaktion lexikographisch sortiert.

Mit einem *Mindestsupport* von 25 Prozent (= 2/8 Transaktionen) werden anschließend anhand einer exemplarischen Anwendung des Apriori Algorithmus die *Large-Itemsets* generiert.

Abbildung 5: Datenbasis.

(k=1) Mengen finden



C_1	Support
A	7
B	3
C	1
D	5
E	1
F	2

L_1
A
B
D
F

Abbildung 6: Generieren der 1-elementigen Large-Itemsets.

Es werden die einzelnen *Items* aus der Menge aller *Items* I herangezogen, um damit die Menge der möglichen 1-elementigen *Large-Itemsets* C_1 zu bilden. Danach wird für jedes Element der *Support* berechnet. Jene Elemente, welche den *Mindestsupport* erreichen, werden in die Menge der 1-elementigen *Large-Itemsets* L_1 übernommen. Im obigen Beispiel werden die Elemente C und E nicht nach L_1 übernommen, da sie nur einen *Support* von 1 haben und damit nicht den *Mindestsupport* von 2 erreichen.

(k=2) Mengen finden

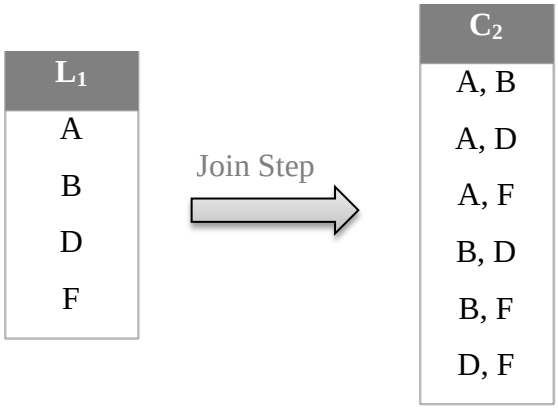


Abbildung 7: Generieren der möglichen 2-elementigen Large-Itemsets.

Aus allen 1-elementigen *Large-Itemsets* L_1 werden über den *Join Step* die Kandidaten der 2-elementigen *Itemsets* C_2 gebildet.

Theoretisch folgt nach dem *Join Step* der *Prune Step*, aber das ist an dieser Stelle noch nicht notwendig, da zu diesem Zeitpunkt klarerweise alle 1-elementigen Teilmengen aus C_2 in L_1 enthalten sind. Der *Prune Step* wird erst ab der Generierung der 3-elementigen Kandidaten benötigt und deshalb erst dort illustriert. (Ester & Sander, 2000, S. 163)

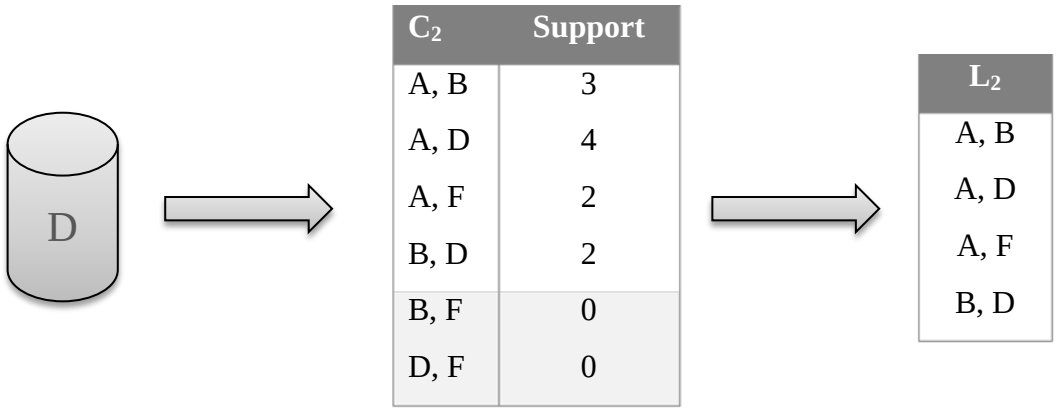


Abbildung 8: Ermitteln der 2-elementigen Large-Itemsets.

Es wird nun die Anzahl der Vorkommnisse der 2-elementigen *Itemsets* aus der Datenbank gelesen um den *Support* zu bestimmen. Die *Itemsets* (B, F) und (D, F) erreichen nicht den *Mindestsupport* von 2 und werden daher nicht in die Menge der 2-ementigen *Large-Itemsets* L_2 übernommen.

(k=3) Mengen finden

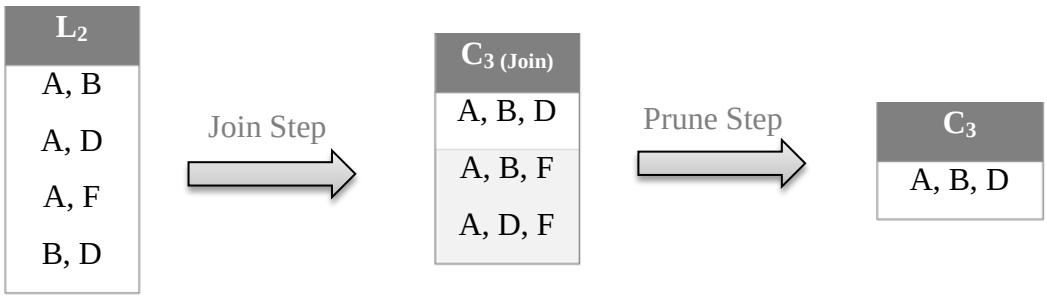


Abbildung 9: Mögliche 3-elementige Large-Itemsets.

Um nun wieder die Kandidaten für *Large-Itemsets* zu erhalten, wird abermals der *Join-Step* ausgeführt. Ein *Join* wird nach folgender Regel durchgeführt: Vorausgesetzt wird, dass die *Itemsets* lexikographisch sortiert sind. Es wird jedes k-elementige *Large-Itemset* p jeweils mit dem letzten Element jener k-elementigen *Large-Itemsets* q verlängert, die in den ersten k-1 Elementen übereinstimmen. (Ester & Sander, 2000, S. 163)

Im gegebenen Beispiel sind in L_2 bei den *Itemsets* {(A, B); (A, D); (A, F)} die ersten k-1 identisch und daher ergeben sich nach dem *Join* die folgenden *Itemsets* $C_{3\ Join}$ = {(A, B, D); (A, B, F); (A, D, F)}

Der *Prune-Step* besagt, dass alle Itemsets entfernt werden, die eine (k-1)-elementige Teilmenge enthalten, welche nicht in den k-1 *Large-Itemsets* vorkommt.

In diesem Beispiel werden die *Itemsets* {(A, B, F); (A, D, F)} im *Prune-Step* entfernt. Aus (A, B, F) können die (k-1)-elementigen Teilmengen {(A, B); (A, F); (B, F)} gebildet werden. Da (B, F) nicht in L_2 enthalten ist kann (A, B, F) aus $C_{3\ Join}$ entfernt werden. (A, D, F) enthält die Teilmengen {(A, B); (A, F); (D, F)}, wobei (D, F) nicht in L_2 enthalten ist. Daher wird dieses *Itemset* ebenfalls aus $C_{3\ Join}$ entfernt.

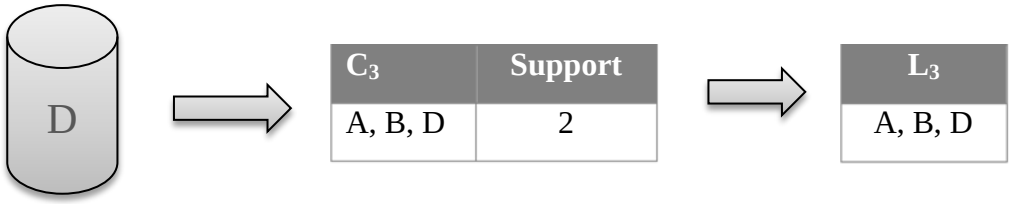


Abbildung 10: Die 3-elementigen Large-Itemsets

Letztendlich wird abermals der *Support* C_3 gegen die Datenbank überprüft und weil das *Itemset* (A, B, D) den *Mindestsupport* erreicht, wird es nach L_3 übernommen. Es können aus C_3 keine 4-elementigen *Itemsets* mehr gebildet werden und die Abbruchbedingung für den Apriori Algorithmus ist erreicht. Das Ergebnis des Beispiels wären L_1, L_2 und L_3.

4.1.3 Der AprioriTid Algorithmus

Der Apriori Algorithmus durchsucht in jeder k-ten Iteration die gesamte Datenbank. Obwohl *Itemsets*, die den *Mindestsupport* nicht erfüllen, in der darauf folgenden Iteration nicht mehr von Bedeutung sind, wird bei jedem Durchlauf die gesamte Datenbank gescannt. Dieser Umstand führt bei einer größeren Datenmenge zu Performanceproblemen. (Petersohn, 2005, S. 111)

Der AprioriTid Algorithmus kann als eine Variante des Apriori gesehen werden, welche sich der Performanceproblematik des mehrfachen Durchsuchens der kompletten Datenbank annimmt. Diese spezielle Variante des Apriori macht nur bei der ersten Iteration einen vollständigen Scan der Datenbasis und versucht nach jedem weiteren Durchlauf die Datenbasis zu verkleinern. Diese verkleinerte Datenbasis wird $\overline{C}_k$ bezeichnet und in den folgenden Iterationen wird nur mehr diese Datenbasis $\overline{C}_k$ zur *Supportermittlung* herangezogen. Die *apriori-gen*() Funktion bleibt unverändert zum Apriori Algorithmus. (Hippner, Küsters, Meyer, & Wilde, 2001, S. 434)

Jede Transaktion in $\overline{C}_k$ hat die Form <TID, $\{X_k\}$>, wobei X_k für ein potentielles *Large k-Itemset* mit der eindeutigen TransaktionsID (kurz TID) steht. Für k=1 entspricht $\overline{C}_1$ der Datenbasis D. Es wird lediglich jedes *Item* i mit dem *Itemset* {i} ersetzt. Für k größer 1 bestehen die Transaktionen in $\overline{C}_k$ nur noch aus den Teilmengen, welche im aktuellen k-*Candidateset* C_k auch vorkommen. Jene Transaktionen, welche danach kein k-*Candidateset* enthalten werden in $\overline{C}_k$ beseitigt. Während der Abarbeitung wird somit einerseits die Anzahl der Transaktionen verkleinert und andererseits kann auch die Anzahl der Einträge in einer Transaktion reduziert werden. Dieser Effekt[6] macht sich jedoch nur für ein großes k bemerkbar. Bei einem kleinen k kann es sogar zu einer gegenteiligen Wirkung kommen, da die Datenbasis anfangs noch sehr groß ist und sich dadurch viele Kombinationsmöglichkeiten ergeben, was zu vielen k-*Candidatesets* führt. (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 490)

[6] Für eine gute Veranschaulichung dieses Effektes möchte ich auf (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 496) verweisen, wo in Figure 6 der Apriori Algorithmus mit dem AprioriTid in den einzelnen Phasen verglichen wird.

Im Folgenden wird die Funktionsweise des AprioriTid in Form von Pseudocode veranschaulicht. Es wird nur der Code AprioriTid ohne die apriori-gen() Funktion dargestellt, da diese Funktion identisch mit der des Apriori Algorithmus ist.

Notation:
L_1: {large 1-itemsets};
$\overline{C}_1$: database D;
for (k=2; $L_{k-1} \neq \emptyset$; k++) **do begin**
    C_k = apriori-gen(L_{k-1}); *// new candidates*
    $\overline{C}_k = \emptyset$;
    forall entries t $\in \overline{C}_{k-1}$ **do begin**
        // determine candidate itemsets in C_k contained
        // in the transaction with identifier t.TID
        $C_t = \{c \in C_k \mid (c - c[k]) \in$ t.set-of-itemsets $\wedge$
            $(c - c[k-1]) \in$ t.set-of-itemsets};
        forall candidates $c \in C_t$ **do**
            c.count++;
        if ($C_t \neq \emptyset$) **then** $\overline{C}_k$ += <t.TID,C_t>;
    end
    $L_k = \{c \in C_k \mid$ c.count $\geq$ minsup$\}$
end
Answer = $\bigcup_k L_k$;

Listing 3: AprioriTid nach (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 491).

Da sich der AprioriTid vom Apriori Algorithmus ausschließlich in der Datenbasis zur Berechnung des *Supports* unterscheidet, werden nachfolgend nur die Änderungen der Datenbasis für dasselbe Beispiel wie bereits beim Apriori Algorithmus dargestellt.

Generieren der Datenbasis $\overline{C}_1$

Für k = 1 ist $\overline{C}_1$ eine Abbildung der Datenbasis. Es werden nur die einzelnen *Items* i mit dem *Itemset* {i} ersetzt. In diesem Schritt wird, das einzige Mal, mit der gesamten Datenbank als Datenbasis gearbeitet.

D

TID	Itemset
T1	A, C, D
T2	D
T3	A, B, D, E
T4	A, F
T5	A, B
T6	A, F
T7	A, D
T8	A, B, D

$\overline{C}_1$

TID	Set of Itemsets
T1	{A}, {C}, {D}
T2	{D}
T3	{A}, {B}, {D}, {E}
T4	{A}, {F}
T5	{A}, {B}
T6	{A}, {F}
T7	{A}, {D}
T8	{A}, {B}, {D}

Abbildung 11: Erstellen der Datenbasis für AprioriTid aus der Datenbank.

Generieren der Datenbasis $\overline{C}_2$

C_2

Itemset	Support
A, B	3
A, D	4
A, F	2
B, D	2
B, F	0
D, F	0

$\overline{C}_2$

TID	Set of Itemsets
T1	{A, D}
T2	
T3	{A, B} {A, D} {B, D}
T4	{A, F}
T5	{A, B}
T6	{A, F}
T7	{A, D}
T8	{A, B} {A, D} {B, D}

Abbildung 12: Erstellen der Datenbasis aus C_2.

Für ein k, das größer als eins ist, werden die Transaktionen in $\overline{C}_k$ mit den *Itemsets* aus C_k gebildet, welche auch in $\overline{C}_{k-1}$ enthalten sind.

Beispielsweise wird {A, B} in $\overline{C}_2$ zu den Transaktionen T3, T5 und T8 hinzugefügt, da genau dieses *Itemset* in $\overline{C}_1$ bei den Transaktionen T3, T5 und T8 zu finden ist. Dies funktioniert analog für alle weiteren *Itemsets* aus C_2. Die *Itemsets* {B, F} und {D, F} können nicht in $\overline{C}_1$ gefunden werden und werden daher auch nicht nach $\overline{C}_2$ übernommen. In $\overline{C}_2$ ist nun in der Transaktion T2 kein einziges *Itemset* mehr enthalten, weshalb diese Transaktion entfernt werden kann.

Generieren der Datenbasis $\overline{C}_3$

C_3

Itemset	Support
A, B, D	2

$\overline{C}_3$

TID	Set of Itemsets
T1	
T3	{A, B, D}
T4	
T5	
T6	
T7	
T8	{A, B, D}

Abbildung 13: Erstellen der Datenbasis aus C_3.

C_3 besteht nun nur mehr aus einem *Itemset*. Dieses *Itemset* ist ausschließlich in den Transaktionen T3 und T8 in $\overline{C}_2$ zu finden und wird daher auch in den Transaktionen T3 und T8 in $\overline{C}_3$ eingetragen. Hier endet zwar der Algorithmus, man kann aber recht deutlich sehen, wie stark die Datenbasis nach lediglich drei Schritten reduziert wurde, letztendlich bleiben nur mehr T3 und T8 übrig. Da der AprioriTid durch die vielen Kombinationsmöglichkeiten der *Items* in frühen Durchläufen eine schlechtere Leistung erbringen kann, wurde versucht auch für diesen Umstand eine Lösung zu finden. Diese wird im folgenden Abschnitt beschrieben.

4.1.4 Der AprioriHybrid Algorithmus

Der AprioriHybrid Algorithmus ist eine Kombination aus dem Apriori und AprioriTid Algorithmus. Vorausgeschickt sei, dass der Apriori Algorithmus dem AprioriTid in den frühen Durchläufen überlegen ist. Diese Erkenntnis macht sich der AprioriHybrid zu Nutzen, indem er bei den anfänglichen Iterationen den Apriori Algorithmus einsetzt und erst später, wenn dessen Performancevorteile zu wirken beginnen, auf den AprioriTid wechselt. (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 496)

Die Performancevorteile des Apriori Algorithmus in frühen Durchläufen werden auf hohe Input-/Output-Kosten zurückgeführt, da die Daten aus $\overline{C}_k$ noch nicht im Arbeitsspeicher gehalten werden können und somit auf die Festplatte ausgelagert werden müssen. Der

Zeitpunkt für den Wechsel von Apriori auf den AprioriTid wird über eine Funktion[7] ermittelt, welche die Größe der Datenbasis $\overline{C}_k$ abschätzt. Sobald $\overline{C}_k$ klein genug ist, um im Arbeitsspeicher gehalten zu werden, wird der Wechsel durchgeführt. (Petersohn, 2005, S. 413)

4.1.5 Der FP-growth Algorithmus

Alle Apriori Versionen haben gemein, dass sie in Zwischenschritten immer *Candidatesets* generieren, was bei langen *Patterns* besonders aufwendig ist. Sind beispielsweise 10^4 einelementige *Large-Itemsets* gegeben werden vom Apriori mehr als 10^7 zweielementige *Candidatesets* generiert und anschließend deren *Support* überprüft. Da Apriori, AprioriTid und AprioriHybrid die *Candidatesets* gleichartig erstellen, bleibt dieser Rechenaufwand keiner dieser Apriori Versionen erspart. (Han, Pei, & Yin, 2000, S. 1)

Der *Frequent Pattern Growth* Algorithmus (kurz *FP-growth*) findet die *Frequent-Itemsets* ohne die Erstellung von *Candidatesets*. Eine Eigenschaft des *FP-growth* Algorithmus ist, dass er die Transaktionen der Datenbasis in einen sogenannten *Frequent-Pattern-Tree* überführt. Dieser Baum ist eine Erweiterung der *Prefix-Tree* Struktur und ermöglicht es einerseits die Datenbasis in Form eines Baumes zu verdichten, um Speicher zu sparen, und andererseits über den *FP-Tree* die *Large-Itemsets* zu finden. (Han, Pei, & Yin, 2000, S. 1)

Laut (Han, Pei, & Yin, 2000, S. 1) lassen sich die Stärken des *FP-growth* Algorithmus durch folgende drei Eigenschaften zusammenfassen:

- Komprimierung der Datenbasis in einen kleineren verdichteten *Frequent-Pattern-Tree* und Vermeidung vieler arbeitsintensiver Datenbankdurchläufe
- Keine Erstellung von *Candidatesets* und die damit verbundenen Überprüfungen ihrer Häufigkeiten
- *Divide and Conquer*: Zerlegung der Aufgaben in kleinere Teilaufgaben und Einschränken des Suchraumes

Nach (Han, Pei, & Yin, 2000, S. 3) lässt sich die Struktur eines *Frequent-Pattern-Tree* wie folgt definieren:

1. Er besteht aus einer Wurzel mit dem Namen „*null*“, einer Menge aus *Sub-Prefix-Trees* als Kinder der Wurzel und einer zusätzlichen *Frequent-Item-Headertable*

[7] Für die genaue Berechnung des Wechselzeitpunktes verweise ich auf (Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 496)

2. Jeder Knoten in einem *Sub-Prefix-Tree* enthält folgende drei Felder:
    - *Item-Name*: der Name des *Items* kennzeichnet, welches *Item* mit diesem Knoten dargestellt wird
    - *Count*: der *Count* steht für die Anzahl der Transaktionen, die diesen Teil des Pfades erreichen
    - *Node-Link*: der *Node-Link* zeigt auf den nächsten Knoten im Baum mit demselben *Item-Name*, sofern ein solcher existiert
3. Die *Headertable* ist aus folgenden zwei Feldern zusammengesetzt:
    - *Item-Name:* dieser ist wie in Punkt 2 definiert
    - *Node-Link*: der *Node-Link* zeigt auf das erste gleichnamige *Item* im Baum und erleichtert das Auffinden aller Knoten eines bestimmten *Items*

4.1.5.1 Erstellung des FP-Tree

Die Erstellung des FP-Tree benötigt exakt zwei Scans der Datenbasis. Im ersten Durchlauf werden die k-1 *Frequent-Itemsets* ermittelt und im zweiten Durchlauf wird der Baum aufgebaut. Der Aufbau des *FP-Tree* wird nachfolgend anhand eines Beispiels aus (Han, Pei, & Yin, 2000, S. 3) demonstriert.

1. Schritt: Ermitteln der k-1 *Frequent-Items*

Gegeben sei die Datenbasis D mit den Items I = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p} und die Festlegung des *Mindestsupports* für *Frequent-Itemsets* mit 3. Die Datenbasis wird einmal durchgegangen um die Menge der *Frequent-Items F* zu bestimmen. Jene *Items* aus *F*, welche den *Mindestsupport* erreichen, werden in absteigender Reihenfolge sortiert, wodurch sich *L* (*List of Frequent-Items*) ergibt.

D

TID	Itemset
T100	f, a, c, d, g, i, m, p
T200	a, b, c, f, l, m, o
T300	b, f, h, j, o
T400	b, c, k, s, p
T500	a, f, c, e, l, p, m, n

F

Item	Sup.
a	3
b	3
c	4
f	4
m	3
p	3

L

Item	Sup.
f	4
c	4
a	3
b	3
m	3
p	3

Abbildung 14: Ermittlung der k-1 *Frequent-Itemsets*.

2. Schritt: Aufbau des Baumes

Die Wurzel des Baumes (*T*) wird erstellt und „*null*“ genannt. Im Anschluss wird für jede Transaktion Folgendes durchgeführt:

1. Es werden nur die *Frequent-Items* aus der Transaktion selektiert und analog zu *L sortiert*.

TID	Itemset
T100	f, a, c, d, g, i, m, p

TID	Itemset
T100	f, c, a, m, p

Abbildung 15: Vorbereiten der Transaktion T100 für den FP-Tree.

2. Die Liste von *Frequent-Items* einer Transaktion sei *[p|P]*, wobei *p* für das erste *Item* und *P* für die restlichen Einträge in der Liste steht. Es wird nun die Funktion *insert_tree([p|P], T)* aufgerufen.
3. Diese Funktion arbeitet folgendermaßen: Falls *T* einen Child-Knoten mit demselben *Item-Name* wie *p* hat, dann wird der *Count* dieses Knotens um 1 inkrementiert. Ansonsten wird ein neuer Knoten *N* mit *Count* = 1 erstellt. Als *Parent-Link* wird *T* eingetragen und über die *Node-Link* Struktur wird ein *Node-Link* für die gleichnamigen *Items* gesetzt. Falls *P* nicht leer ist wird die Funktion *insert_tree(P, N)* rekursiv weiter aufgerufen.

Nachdem die erste Transaktion mit der TID = T100 abgearbeitet ist sehen der Baum und die Header Table folgendermaßen aus.

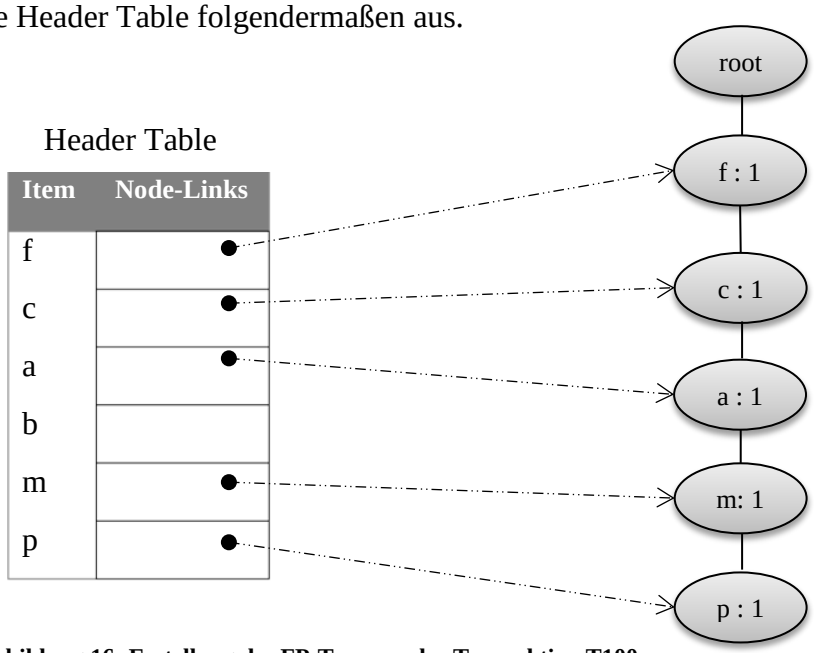


Abbildung 16: Erstellung des FP-Tree aus der Transaktion T100.

Aus der Transaktion T200 werden ebenfalls nur jene *Item*s, die auch in L vorkommen selektiert und analog zu L sortiert.

TID	Itemset
T200	a, b, c, f, l, m, o

TID	Itemset
T200	f, c, a, b, m

Abbildung 17: Vorbereiten der Transaktion T200 für den FP-Tree

Die ersten *Items* der Transaktion {f, c, a} sind bereits in derselben Reihenfolge im Baum vorhanden und es müssen nur die Counter für diese Elemente inkrementiert werden. Das *Item* {a} hat noch kein Kind {b}, weshalb ein neuer Zweig im Baum entsteht. Die *Counter* der neuen *Items* werden mit 1 initialisiert. Da {b} noch nicht in der *Headertable* vorhanden ist, wird in dieser der *Node-Link* gesetzt. Auch für {m} wird ein *Node-Link* vom letzten *Item* in der Node-Linkkette gesetzt.

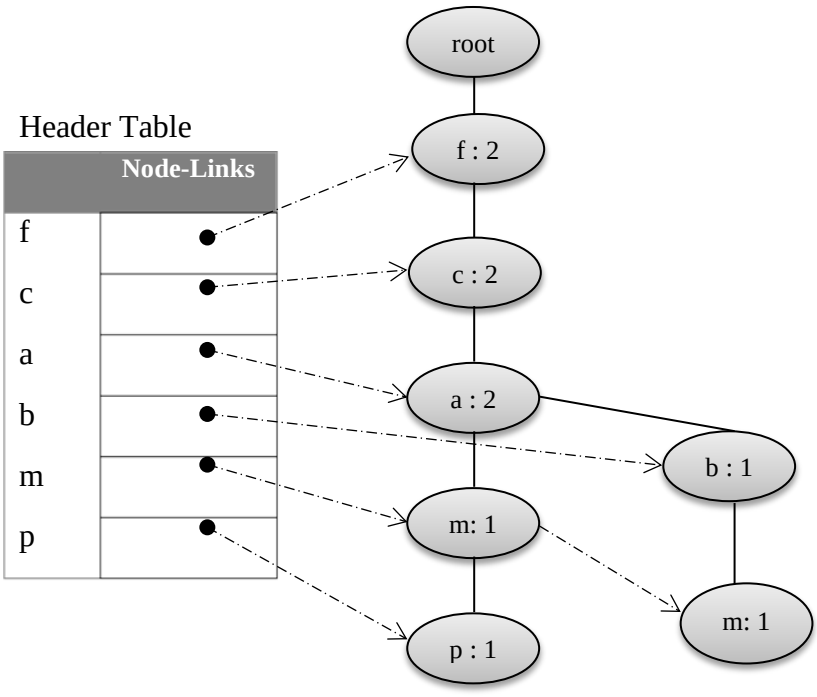


Abbildung 18: FP-Tree nach Transaktion T200.

Die Abarbeitung der nächsten Transaktionen T300, T400 und T500 erfolgt nach demselben Schema und wird deshalb nicht zusätzlich erklärt. Es wird lediglich das Endergebnis in Abbildung 19 dargestellt.

Nachdem die Datenbank ein zweites Mal durchgegangen wurde und alle Transaktionen berücksichtigt wurden wäre der folgende Baum das Resultat.

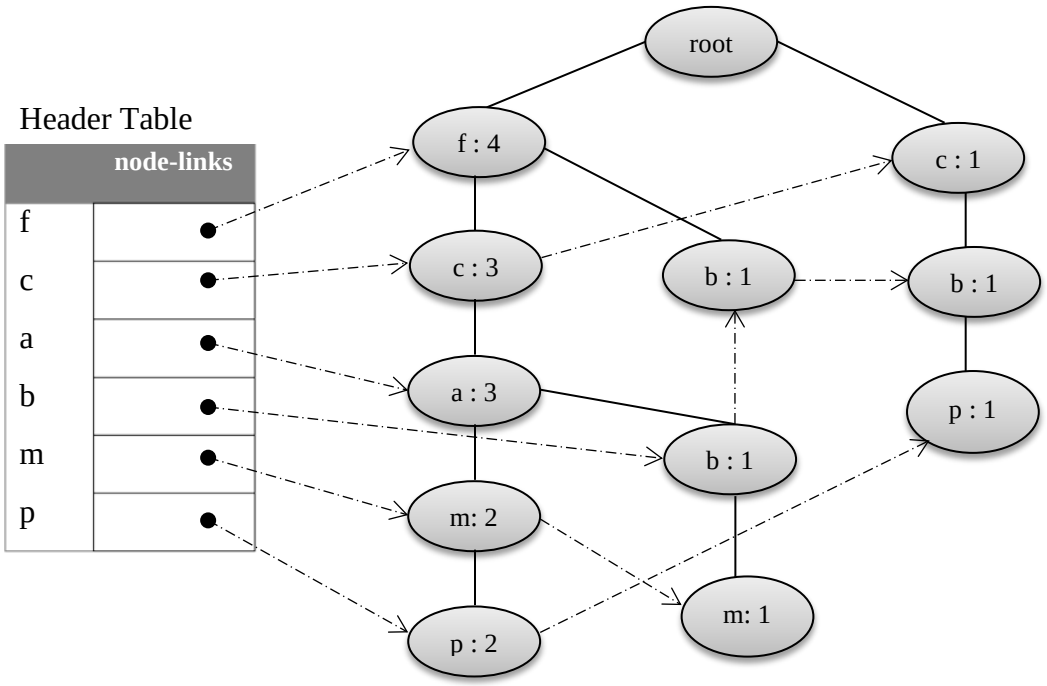


Abbildung 19: FP-Tree angelehnt an Beispiel aus (Han, Pei, & Yin, 2000, S. 3).

4.1.5.2 Extrahieren von Large-Itemsets aus dem FP-Tree

Die vorangegangene Konstruktion der kompakten *FP-Tree* Datenstruktur ist die Voraussetzung, dass die *Frequent-Patterns* extrahiert werden können. In diesem Abschnitt wird anhand des Beispiels erläutert, wie die *Frequent-Patterns* effizient aus der Baumstruktur gewonnen werden können. Die folgenden zwei Eigenschaften, welche in (Han, Pei, & Yin, 2000, S. 5ff) definiert wurden, sind wesentlich für die Erzeugung der *Frequent-Patterns*.

***Node-Link Property*:** Für ein häufiges *Item* a_i können alle *Frequent-Patterns*, welche a_i beinhalten, durch das Folgen des *Node-Link* gewonnen werden. Begonnen wird in der *Header Table*, da diese alle *Frequent-Items* beinhaltet. (Han, Pei, & Yin, 2000, S. 5)

Prefix-Path Property: Um die *Frequent-Patterns* für einen Knoten a_i in einem Pfad P zu berechnen wird nur der Prefix-Pfad des Knoten a_i in *P* berücksichtigt. Des Weiteren bekommen alle Knoten des Prefix-Pfades denselben *Count* wie der Knoten a_i. (Han, Pei, & Yin, 2000, S. 6)

Die Extrahierung der *Frequent-Patterns* wird im Folgenden anhand des erstellten FP-Tree aus Abbildung 19 demonstriert. Begonnen wird am Ende der *Headertable,* welche von unten

nach oben abgearbeitet wird. Basierend auf der *Prefix-Path Property* wird für jedes *Item* eine sogenannte C*onditional Pattern Base* gebildet. Aus dieser wird dann für jedes *Item* ein *Conditional-FP-Tree* gebildet und aus diesen Unterbäumen können letztendlich durch die Kombination der Knoten die *Frequent-Patterns* generiert werden. Der Pseudocode zur Extrahierung der *Frequent-Patterns* ist in Listing 4 demonstriert.

```
Procedure FP-growth( Tree, α)
{
        if  Tree contains a single Path P
                then for each combination (denoted as β) of the nodes in the path P do
                        generate pattern β ∪α with  = minsup of nodes in β;
        else for each a_i in the header of Tree do
        {
                generate pattern β = a_i ∪α with sup = a_i.sup;
                construct β's conditional pattern base and
                The β's conditional  FP-tree Tree_β;
                if Tree_β ≠ ∅
                then call FP-growth (Tree_β, β)
        }
}
```

Listing 4: Ermittlung von Large-Itemsets aus FP-Tree nach (Han, Pei, & Yin, 2000, S. 7).

Begonnen wird mit dem *Item p*, da es das letzte *Item* in der *Headertable* ist. Entsprechend der zuvor beschriebenen *Node-Link Property*, folgt man ausgehend von der Headertable dem *Node-Link*. Für p können zwei Pfade im Baum gefunden werden. Dies sind für p die Pfade (f:4, c:3, a:3, m:2, p:2) und (c:1, b:1, p:1). Diese beiden Pfade sind die Grundlage für die *Conditional Pattern Base* von *p*. Wie in der *Prefix-Path-Property* beschrieben werden die Prefix-Pfade, ausgehend von *p*, berücksichtigt und die *Counter* der einzelnen *Prefixitems* erhalten denselben *Count* wie *p*. Dies ergibt die folgenden Pfade als *Conditional Pattern Base*: (f:2, c:2, a:2, m:2) und (c:1, b:1). Diese *Conditional Pattern Base* dient nun als Grundlage für die Erstellung des Conditional FP-Tree. Nach diesen zwei Schritten wird mit der *Conditional Pattern Base* einfach ein eigener FP-Tree nach demselben Schema wie zuvor

erstellt. Da nur das *Item* c den *Minimumsupport* von 3 erfüllt, besteht der Baum nur aus diesem Element. Der *Conditional Tree* wird in der Form „FP-Tree | a_i“ angegeben. Für das *Item* p wäre damit der *Conditional Tree* {(c:3)} | p. (Han, Pei, & Yin, 2000, S. 5ff)

Nach demselben Schema werden die restlichen *Items* von unten nach oben durchgegangen und jeweils die *Conditional Pattern Base* und der *Conditional Tree* erstellt. Die Ergebnisse für jedes *Item* sind in Tabelle 2 ersichtlich.

Item	Conditional Pattern Base	Conditional FP-Tree
p	{(f:2, c:2, a:2, m:2), (c:1, b:1)}	{(c:3)} | p
m	{(f:2, c:2, a:2), (f:1, c:1, a:1, b:1)}	{(f:3, c:3, a:3)} | m
b	{(f:1, c:1, a:1), (f:1), (c:1)}	∅
a	{(f:3, c:3)}	{(f:3, c:3)} | a
c	{(f:3)}	{(f:3)} | c
f	∅	∅

Tabelle 2: Conditional Pattern Base und FP-Tree aller Frequent-Items.

Nachdem die *Conditional FP-Trees* erstellt wurden, können die *Large-Itemsets* generiert werden, indem das jeweilige *Item* alle möglichen Kombinationen mit seinem *Conditional FP-Tree* bildet. Dies würde letztendlich die *Frequent-Patterns* ergeben, welche in Tabelle 3 aufgelistet sind.

Item	Conditional FP-Tree	Frequent-Patterns
p	{(c:3)} | p	{p, cp}
m	{(f:3, c:3, a:3)} | m	{m, fm, cm, am, fcm fam, cam, fcam}
b	∅	{b}
a	{(f:3, c:3)} | a	{a, fa, ca, fca}
c	{(f:3)} | c	{c, fc}
f	∅	{f}

Tabelle 3: Frequent-Patterns aus den Conditional FP-Trees.

4.1.6 Regelgenerierung

Es wurde bereits veranschaulicht, wie mit den *Apriori Versionen* oder dem *FP-growth* Algorithmus jene *Large-Itemsets* extrahiert werden, deren jeweiliger *Support* dem *Mindestsupport* entspricht. Im nächsten Schritt werden aus den *Large-Itemsets* Regeln abgeleitet. Aus jedem *Large-Itemset* l_k aus L_k mit $k > 1$ werden Regeln der Form $X \rightarrow l_k - X$ gebildet, wobei $X \subset l_k$ und $X \neq \emptyset$. (Hippner, Küsters, Meyer, & Wilde, 2001, S. 435)

Im Gegensatz zum *Support* hat die *Confidence* keine Monotonieeigenschaft. Es kann beispielsweise die *Confidence* für $X \rightarrow Y$ größer, kleiner oder gleich sein als die *Confidence* für $\overline{X} \rightarrow \overline{Y}$, wobei $\overline{X} \subseteq X$ und $\overline{Y} \subseteq Y$. Dennoch kann für Regeln, die aus demselben *Large-Itemset* l_k erstellt wurde, folgendes Theorem aufgestellt werden: (Tan, Steinbach, & Kumar, 2005, S. 350)

- Erfüllt eine Regel $X \rightarrow l_k - X$ nicht die geforderte *Mindestconfidence*, dann kann davon ausgegangen werden, dass jede Regel $X' \rightarrow l_k - X'$ ebenfalls nicht die *Mindestconfidence* erreichen wird, wobei X‘ eine Teilmenge von X ist. (Tan, Steinbach, & Kumar, 2005, S. 350)

Um dieses Theorem zu bestätigen müssen lediglich die folgende zwei Regeln betrachtet werden: $X' \rightarrow l_k - X'$ und $X \rightarrow l_k - X$, wobei $X' \subset X$. Die *Confidence* der beiden Regeln errechnet sich über $support(l_k) / support(X)$ und $support(l_k) / support(X')$. Da X‘ eine Teilmenge von X ist gilt $support(X') \geq support(X)$. Daraus ergibt sich, dass die erste Regel unmöglich eine höhere *Confidence* als die zweite Regel haben kann. (Tan, Steinbach, & Kumar, 2005, S. 350)

Wegen diesem Theorem werden sukzessive für jedes *Large-Itemset* (l_k) aus L_k (mit $k > 1$) separat alle Regeln generiert. Als Information zum Verständnis der nachfolgenden Erklärungen sei erwähnt, dass das Vorderglied der Regel Antezedens und das Hinterglied Konsequenz genannt wird.

Im ersten Schritt werden alle Regeln gebildet, die in der Konsequenz ein *Item* enthalten. Aus jenen *Items*, welche die angegebene *Mindestconfidence* erfüllen werden mit der bereits bekannten *apriori-gen* Funktion alle Regeln mit zwei *Items* in der Konsequenz gebildet und wieder die *Mindestconfidence* überprüft. Dies wird rekursiv mit den Konsequenzen, deren Regeln die *Mindestconfidence* erreichen, fortgeführt, bis für l_k keine weiteren Regeln mehr gebildet werden können. In Listing 5 wird der Algorithmus zum Generieren der Regeln aus

(Agrawal & Srikant, Fast Algorithms for Mining Assiciation Rules, 1994, S. 317) in Form von Pseudocode demonstriert.

```
forall large k-itemsets l_k ∈ L_k, k ≥ 2 do begin
    H_1 = {consequents of rules from l_k with one item in the consequent};
    call ap-genrules(l_k, H_1);
end

procedure ap-genrules(l_k: large k-itemset, H_m: set of m-item consequents)
    if (k > m + 1) then begin
        H_m+1 = apriori-gen(H_m);
        forall h_m+1 ∈ H_m+1 do begin
            conf = support(l_k) / support(l_k – h_m+1);
            if (conf ≥ confmin) then
                output the rule (l_k – h_m+1) → h_m+1
                    with confidence = conf and support = support(l_k);
            else
                delete h_m+1 from H_m+1
        end
        call ap-genrules(l_k, H_m+1);
    end
```

Listing 5: Algorithmus zur Erzeugung von Assoziationsregeln.

Nachfolgend wird die Arbeitsweise des Algorithmus anhand der *Large-Itemsets* aus dem Beispiel von Kapitel 4.1.2 veranschaulicht. Die folgenden *Large-Itemsets* L_k werden zur Regelgenerierung herangezogen. Die *Mindestconfidence* wird mit 60 Prozent festgelegt.

L_1	Support
A	7
B	3
D	5
F	2

L_2	Support
A, B	3
A, D	4
A, F	2
B, D	2

L_3	Support
A, B, D	2

Abbildung 20: Large-Itemsets als Basis zur Regelgenerierung.

Im ersten Schritt werden aus L_2 für jedes *Itemset* l_2 alle Regeln mit 1-elementigen Konsequenzen gebildet und für jede Regel die *Confidence* bestimmt. Die Konsequenzen der Regeln, welche die *Mindestconfidence* erreichen, werden zu H_1 hinzugefügt. Für $l_2 = \{A, B\}$ lassen sich die Regeln $B \rightarrow A$ und $A \rightarrow B$ erstellen. Da nur $B \rightarrow A$ die *Mindestconfidence* erreicht wird auch nur deren Konklusion zu H_1 hinzugefügt. Es wird die *ap-genrules()* Prozedur aufgerufen aber für die 2-elementigen *Itemsets* werden keine weiteren Schritte mehr ausgeführt, da keine weiteren Kombinationsmöglichkeiten bestehen. Im Folgenden sind die erstellten Regeln aus den 2-elementigen *Itemsets* dargestellt:

Regel	Konklusion	Confidence	H_1	gültige Regeln
$B \rightarrow A$	A	1,00	{A}	{B → A}
$A \rightarrow B$	B	0,43	{A}	
$D \rightarrow A$	A	0,80	{A}	{D → A}
$A \rightarrow D$	D	0,57	{A}	
$F \rightarrow A$	A	1,00	{A}	{F → A}
$A \rightarrow F$	F	0,29	{A}	
$D \rightarrow B$	B	0,40	∅	{B → D}
$B \rightarrow D$	D	0,67	{D}	

Tabelle 4: Regeln aus 2-elementigen Itemsets.

In der nächsten Stufe werden aus L_3 mit dem einzigen enthaltenen *Itemset* $l_3 = \{A, B, D\}$ wieder alle Regeln mit 1-elementigen Konklusionen gebildet. Dies ergibt die in Tabelle 5 enthaltenen Regeln:

Regel	Konklusion	Confidence	H_1	gültige Regeln
B, D → A	A	1,00	{A}	{B, D → A} {A, B → D}
A, D → B	B	0,50	{A}	
A, B → D	D	0,67	{A, D}	

Tabelle 5: Regeln aus 3-elementigen Itemsets.

Nach dem Bilden der 1-elementigen Regeln wird mit $H_1 = \{A, D\}$ und $l_k = \{A, B, D\}$ die *ap-genrules()* Prozedur aufgerufen. In dieser Funktion werden aus H_1 alle Kombinationen für 2-elementige Konsequenzen, genannt H_2, gebildet. Es gibt nur eine Kombination was zur Regel

B → A, D führt. Nachdem die *Confidence* dieser Regel bei 0,67 liegt und somit die *Mindestconfidence* erfüllt wird diese ebenfalls übernommen. Der Algorithmus terminiert, da keine weiteren Regeln gebildet werden können.

4.1.7 Interessantheit von Assoziationsregeln

Die zuvor generierten Assoziationsregeln wurden anhand des *Support*s und der *Confidence* ermittelt. Die Stärke einer gewonnen Assoziationsregel kann alleine mit diesen zwei Kennzahlen nicht ausreichend bestimmt werden, da bei der Ermittlung der *Confidence* die Wahrscheinlichkeit der Konklusion der Regel nicht beachtet wird. Ein Beispiel aus (Ester & Sander, 2000, S. 168) verdeutlicht diesen Umstand:

Ein Anbieter von Schokoriegeln beobachtet das Verhalten von 5000 Schülern morgens in einer Schule. Aus der Beobachtung ergeben sich folgende Daten:

- 3000 Schüler spielen Fußball (60 Prozent)
- 3750 Schüler essen Schokoriegel (75 Prozent)
- 2000 Schüler machen beides (40 Prozent)

Mit einem minimalen *Support* von 40 Prozent und einen *Confidenceschwellwert* von 60 Prozent ließe sich die Assoziationsregel „spielt Fußball“ → „isst Schokoriegel“ erstellen. Die *Confidence* dieser Regel wäre 67 Prozent, jedoch wäre die Regel irreführend, da der Prozentsatz der Schüler, die Schokoriegel essen, mit 75 Prozent höher ist. In Wirklichkeit gibt es eine negative Korrelation zwischen den Aktivitäten „spielt Fußball“ und „isst Schokoriegel“. Wenn ein Schüler Fußball spielt verringert sich die Wahrscheinlichkeit, dass dieser auch Schokoriegel isst.

Eine solche Regel sollte wieder entfernt werden, da eine Fehlinterpretation die Folge wäre. In (Ester & Sander, 2000, S. 168) wird zum Herausfiltern solcher irreführender Regeln auf einen Ausdruck verwiesen, der als Maß für die „Interessantheit“ einer Regel dienen soll. Für eine Regel, die in der Form „A → B“ vorliegt, wird die Interessantheit mit folgendem Ausdruck bestimmt:

$$\frac{\boldsymbol{P(A \wedge B)}}{\boldsymbol{P(A)}} - \boldsymbol{P(B)} > \boldsymbol{d}$$

Für eine Regel muss damit gelten, dass das Interessantheitsmaß aus dem obigen Ausdruck größer einer festgelegten Konstante *d* sein muss und $d > 0$ ist. Je größer dieser Wert ist umso größer ist auch der Zusammenhang zwischen A und B.

Es gibt eine ganze Fülle an unterschiedlichen Interessantheitsmaßen[8], mit welchen sich Assoziationsregeln bewerten lassen. Die Berechnung und Eigenschaften der Maße *Lift* und die *Conviction* werden noch erwähnt, da diese auch in der Implementierung des Prototyps benutzt wurden.

Lift:

Der *Lift* stellt das gemeinsame Vorkommen von A und B mit dem zu erwartenden Vorkommen bei statistischer Unabhängigkeit gegenüber. Er vergleicht somit, um wie viel sich die Verteilung von *Items* in einer Teilmenge in der Verteilung der Grundgesamtheit unterscheidet (Hippner, Küsters, Meyer, & Wilde, 2001, S. 447). Der *Lift* wird mit folgendem Ausdruck bestimmt:

$$\boldsymbol{lift}(\boldsymbol{A} \rightarrow \boldsymbol{B}) = \frac{\boldsymbol{conf}(\boldsymbol{A} \rightarrow \boldsymbol{B})}{\boldsymbol{sup}(\boldsymbol{B})} = \frac{\boldsymbol{conf}(\boldsymbol{B} \rightarrow \boldsymbol{A})}{\boldsymbol{sup}(\boldsymbol{A})} = \frac{\boldsymbol{P}(\boldsymbol{A} \wedge \boldsymbol{B})}{\boldsymbol{P}(\boldsymbol{A})\boldsymbol{P}(\boldsymbol{B})}$$

Das Ergebnis des *Lift* (A → B) lässt damit folgende Rückschlüsse zu:

- > 1: A und B korrelieren positiv
- = 1: es gibt keine Korrelation zwischen A und B
- < 1: A und B korrelieren negativ

Unter bestimmten Umständen weißt der *Lift* Schwächen auf. Diese können über das Maß der der *Conviction* ausgeglichen werden. Ein solcher Umstand wird bei der Erläuterung der *Conviction* anhand eine Beispiels dargestellt.

Conviction:

Die *Conviction* ist ein weiteres Interessantheitsmaß und versucht, einige Schwächen des *Lift* und der *Confidence* auszugleichen. Im Gegensatz zum *Lift* unterscheidet sich die *Conviction* auch bei umgekehrter Regelrichtung (conv(A → B) ≠ conv(B → A)). Die *Conviction* hat einen Wertebereich von [0.5,…, 1,…∞] und ist, falls A und B unabhängig voneinander sind, wie beim *Lift* 1. Werte, die viel größer als 1, sind deuten auf interessante Regeln hin. Die

[8] Für eine umfangreiche Zusammenfassung einer Vielzahl an Interessantheitsmaßen sei auf (Hahsler, 2011) verwiesen.

Conviction besitzt außerdem die Eigenschaft, bei Regeln mit einer *Confidence* von 100 Prozent, einen Wert von ∞ zu haben. (Azevedo & Jorge, 2007, S. 512)

Berechnet wird die *Conviction* über folgenden Ausdruck:

$$\boldsymbol{conviction}(\boldsymbol{A} \rightarrow \boldsymbol{B}) = \frac{(\mathbf{1} - \boldsymbol{sup}(\boldsymbol{B}))}{(\mathbf{1} - \boldsymbol{conf}(\boldsymbol{A} \rightarrow \boldsymbol{B}))} = \frac{\boldsymbol{P}(\boldsymbol{A})\, \boldsymbol{P}(\neg \boldsymbol{B})}{\boldsymbol{P}(\boldsymbol{A} \wedge \neg \mathbf{B})}$$

Ein Beispiel aus (Hippner, Küsters, Meyer, & Wilde, 2001, S. 447) soll verdeutlichen, wie Schwächen des *Lift* über die *Conviction* ausgeglichen werden.

- 5 Prozent der Bevölkerung sind Vietnamveteranen (alle älter als 5 Jahre)
- 90 Prozent der Bevölkerung sind älter als 5 Jahre.

Der *Lift* mit diesem Beispiel wäre bei (0,05 / 0,05*0,9) = 1,1. Dieses Ergebnis liegt nur knapp über den Wert 1, welcher auf keine Korrelation hinweist. Die *Conviction* hingegen wäre bei ∞ und würde der totalen Abhängigkeit gerecht werden.

4.2 Ausreißererkennung in Transaktionsdaten

In der praktischen Umsetzung wurden die gewonnen Regeln für den Prototyp noch weiterverarbeitet. Diese Weiterverarbeitung basiert auf dem Ansatz der Ausreißererkennung in Transaktionsdaten. Nachfolgend werden zwei Methoden hierzu vorgestellt.

4.2.1 Ausreißererkennung anhand von Frequent-Patterns

In (He, Huang, & Deng, 2005) wird eine Methode behandelt, die verspricht, Ausreißer in einer Datenbasis mit Transaktionsdaten zu erkennen. Mit dieser Methode sollen Transaktionen aufgefunden werden, die sich merklich vom Rest der Transaktionen unterscheiden.

Das Auffinden der abweichenden Transaktionen erfolgt in dieser Methode lediglich anhand der Menge aller *Frequent-Patterns*. In (He, Huang, & Deng, 2005, S. 107) werden die *Frequent-Patterns* einer Datenbasis D, die einen minimalen *Support minsup* unterstützen, als FPS (D, minsup) abgekürzt. Die Überlegungen zur Erkennung der Ausreißer basieren auf folgenden drei Definitionen aus (He, Huang, & Deng, 2005, S. 107):

Definition 1: Ein Ausreißermaß für eine Transaktion t ist der FPOF (*Frequent Pattern Outlier Factor*), der mit folgendem Ausdruck bestimmt wird.

$$\boldsymbol{FPOF(t)} = \frac{\{\boldsymbol{X}|\boldsymbol{X} \subseteq \mathbf{t} \wedge \in \mathbf{FPS(D, minsup)}\} \sum \boldsymbol{support(X)}}{|\boldsymbol{FPS(D, minsup)}|}$$

Zuerst wird die Summe der *Supportwerte* jener *Frequent-Patterns* gebildet, die auch in der Transaktion t enthalten sind. Dieses Ergebnis wird danach durch die Anzahl der *Frequent-Items* geteilt. Der FPOF-Wert liegt zwischen 0 und 1. Transaktionen mit kleinen FPOF-Werten tendieren dazu, Ausreißer zu sein.

Definition 2: Zusätzlich zum FPOF-Wert wird noch beschrieben, warum eine Transaktion eine Abweichung darstellt. Für jede Transaktion t wird ein *Itemset* X aus FPS (D, minsup) als widersprüchlich zu t definiert, wenn $X \not\subset t$ ist. Ein Maß für die Widersprüchlichkeit eines *Itemsets* X zu t wird als *Contradict-ness*(X, t) bezeichnet und durch folgenden Ausdruck bestimmt:

$$\boldsymbol{Contradict - ness(X, t)} = (|\boldsymbol{X}| - |\boldsymbol{t} \cap \boldsymbol{X}|) * \boldsymbol{support(X)}$$

Eine Überlegung hinter dem Ausdruck ist, dass das Maß der Widersprüchlichkeit für ein *Itemset* X auch von dessen *Support* abhängig ist und sich aus einem höheren *Support* auch eine höhere Widersprüchlichkeit ergeben sollte. Eine weitere Überlegung ist, dass längere *Itemsets* eine bessere Beschreibung geben. Daher haben auch diese *Itemsets* einen erhöhenden Einfluss auf das Maß der Widersprüchlichkeit. Die Menge der widersprüchlichen *Itemsets* wird schnell sehr groß und damit nicht überschaubar. Es mach daher Sinn, nur die top k widersprüchlichen *Itemsets* anzuzeigen was zur letzten Definition führt:

Definition 3: TKCFP (*Top k Contradict Frequent Pattern*). Ein *Itemset* X ∈ FPS (D, minsup) ist ein TKCFP, wenn keine weiteren (k-1) *Itemsets* mehr existieren, deren Widersprüchlichkeitsmaß größer ist als jenes von X.

4.2.2 Ausreißererkennung anhand der Assoziationsregeln

Die Methode von (Narita & Kitagawa, 2008) verspricht ebenfalls, Ausreißer in Transaktionsdaten zu finden. Die Bewertung, welche Transaktionen Ausreißer darstellen, unterscheidet sich in dieser Methode von jener aus (He, Huang, & Deng, 2005). Die Basis, nach der die Ausreißer beurteilt werden, ist in diesem Ansatz eine Menge von Assoziationsregeln. Dieses Ausreißermaß definiert eine Transaktion t als Ausreißer, wenn in t *Itemsets* nicht vorhanden sind, obwohl diese laut den Assoziationsregeln eine starke Abhängigkeit zu den *Itemsets* in t haben. Zur Detektion von Ausreißern werden nur Assoziationsregeln mit einer hohen *Confidence* herangezogen. In (Narita & Kitagawa, 2008)

werden nur Regeln mit einer *Confidence* über 80 Prozent herangezogen und zur Bestimmung von abweichenden Transaktionen folgende Definitionen festgelegt:

Definition 1: *Unobserved Rule*: Gegeben ist eine Assoziationsregel in der Form $X \rightarrow Y$. Angenommen es existiert ein *Itemset* $Z \subseteq I$ und für die gegeben Assoziationsregel gilt $X \subseteq Z \wedge Y \notin Z$, dann verletzt Z die Assoziationsregel $X \rightarrow Y$. Eine solche Regel wird auch eine „unbeobachtete Regel von Z“ genannt.

Das Interesse dieser Methode liegt genau in jenen Transaktionen t, in denen mehrere *Items* trotz starker Abhängigkeit fehlen. Um ein Maß für die Abweichung zu erstellen, sollte die ideale Form einer Transaktion t gefunden werden. Diese ideale Form wird auch als t+ bezeichnet und verletzt keine Assoziationsregel. Die Bestimmung von t+ führt zur nächsten Definition:

Definition 2: *Associative Closure*: t+ wird als *Associative Closure* bezeichnet. Für ein *Itemset* $t \subseteq I$ und einer Menge an Regeln R mit einem hohen *Confidence*-Wert wird t+ wie folgt bestimmt:

$$\boldsymbol{t^0 = t}$$

$$\boldsymbol{t^{i+1} = t^i \cup \{e | e \in Y \wedge X \subseteq t^i \wedge X \rightarrow Y \in R\}}$$

$$\boldsymbol{t^+ = t^\infty}$$

In einer Transaktion t steigt die Anzahl der *Items* in t^{i+1}, wenn t^i unbeobachtete Regeln hat. Sobald t^i keine unbeobachteten Regeln mehr enthält, konvergiert t^{i+1} und wird zur *Associative Closure* t^+. Hat eine Transaktion t viele unbeobachteten Regeln, dann ist der Unterschied zwischen t^+ und t dementsprechend größer. Wenn t wenige unbeobachtete Regeln hat, sind sich t und t^+ ähnlicher. Das Maß zur Erkennung von Ausreißern wird in der nächsten Definition bestimmt:

Definition 3: *Outlier Degree*: Das *Outlier Degree* ist das Maß für die Bestimmung von Ausreißern einer Transaktion t (kurz od(t)) und wird über folgende Formel errechnet:

$$\boldsymbol{od(t) = \frac{|\, t^+ - t|}{|\, t^+ |}}$$

Die Eigenschaften dieses Maßes sind: $0 \leq od(t) < 1$. Hatte eine Transaktion keine unbeobachteten Regeln dann gilt $t^+ = t$ und $od(t) = 0$. Der Wert von 1 könnte nur erreicht werden, wenn $t = \emptyset$ ist, was nicht zu erwarten ist. Regeln mit 100 Prozent *Confidence* können ignoriert werden, da diese keine unbeobachteten Regeln sein können.

Definition 4: Als Ausreißer Transaktionen werden jene definiert, für welche od(t) ≥ min_od gilt. Um abweichende Transaktionen zu finden, wird ein Schwellwert für das *Outlier Degree* definiert.

5 Praktische Umsetzung

Die Vorgansweise sowohl bei der Auswahl der Methoden als auch bei der Entwicklung des Prototyps lehnt sich an den CRISP-DM Prozess an. Der Prozess wird nur als Orientierungshilfe bei der Herangehensweise an eine Data Mining Problemstellung gesehen. Es werden nicht alle Prozessschritte vollständig umgesetzt, da der Fokus der Arbeit vor allem in der technischen Umsetzung liegt. Weiters ist eine detaillierte Abarbeitung aller Schritte des CRSIP-DM Prozesses nicht das Ziel und würde den Rahmen der vorliegenden Arbeit sprengen.

5.1 Business Understanding

5.1.1 Unternehmensziele:

Aus der Sicht des Unternehmens besteht die Annahme, dass in der großen Menge von Kalibrierdaten ungenutzte Informationen existieren müssen. Diese Informationen sollen extrahiert werden, um sich einen Wettbewerbsvorteil gegenüber der Konkurrenz sichern zu können.

Die Hauptanwender der Software AVL CRETA™ sind Applikateure von Motorsteuergeräten, welche täglich neu kalibrierte Parameter in dieses System abgeben und diese damit verwalten. Sie haben das entsprechende Domänenwissen zu den Kalibrierdaten und liefert in der Regel den meisten Input für die Umsetzung neuer Funktionen der Software. Viele dieser Applikateure stehen der Idee, aus den Daten nicht-bekannte Informationen zu erhalten und damit einen Mehrwert zu generieren, zwar sehr interessiert gegenüber, zweifeln jedoch zum Teil an der Umsetzbarkeit dieses Vorhabens.

Aus unternehmerischer Sicht ist es daher das Ziel, diese Anwender vom Potential des Data Mining zu überzeugen, damit durch deren Rückmeldungen wiederum neue und nützliche Funktionen entstehen können. Des Weiteren besteht die Hoffnung, dass die Erkenntnisse aus diesem Projekt auch einen Nutzen für andere Bereiche im Unternehmen bringen. Einer der folgenden Anwendungsfälle soll umgesetzt werden:

- Ausreißer-Erkennung, wenn beispielsweise ein Parameter falsch appliziert wurde
- Durch eine automatische Vorberatung soll der Kalibrieraufwand verringert werden
- Falschabgaben von Kalibrierdaten sollen automatisch erkannt werden (es wurden Kalibrationen nicht in allen vorgesehenen Variantenzweigen abgegeben)

5.1.2 Data Mining Ziele

Ein Prototyp soll implementiert werden, welcher einen Anwendungsfall umsetzt und damit beweist, dass der Einsatz von Data Mining Technologien in dieser Domäne umsetzbar ist. Die Funktionsfähigkeit des Prototyps soll anhand eines realen Applikationsprojektes, das von der AVL GmbH bereitgestellt wird, demonstriert werden.

5.1.3 Ausgangssituation

AVL CRETA™ verwaltet alle Daten, die während des Kalibrierprozesses anfallen und ist in C Sharp und C implementiert. Das Tool unterstützt die Datenbanksysteme Oracle Database, Microsoft SQL Server und Microsoft Access, wobei letzteres nicht für ein gesamtes Kalibrierprojekt, sondern nur zur Applizierung von Teilbereichen gedacht ist, beispielsweise während Erprobungsfahrten in abgelegene Gebiete. Diese Daten werden, nachdem die Erprobungsfahrt abgeschlossen ist, wieder in den Datenbankserver eingespielt.

Damit kein zusätzlicher Installationsaufwand beim Kunden entsteht, soll der Prototyp ebenfalls dieselben Datenbanksysteme unterstützen. Eine Unterstützung von Microsoft Access ist ebenfalls erwünscht, damit die Funktionsweise des Prototyps, auch ohne eine Verbindung zu einem Datenbankserver bei Kundenpräsentationen demonstriert werden kann. Das benutzen von Open Source Tools ist nicht erwünscht, da dies ein Kaufhindernis für mehrere Kunden darstellt. Der Prototyp soll in C Sharp geschrieben werden, da dafür alle Softwarelizenzen vorhanden sind. Als Schlüsselpersonen und Ansprechpartner stehen der Produktmanager von CRETA sowie Applikateure mit dem entsprechenden Domänenwissen unterstützend zur Verfügung.

5.2 Data Understanding

In diesem Abschnitt wird die allgemeine Struktur der Daten beschrieben, die für das Data Mining herangezogen werden sollen. Um ein grundlegendes Verständnis zum Aufbau der Daten und der Arbeitsweise der Software AVL CRETA™ zu geben, wird mit dem Tool ein beispielhafter Kalibrierprozess skizziert. Des Weiteren werden in diesem Abschnitt die zu erwartenden Datenmengen aufgezeigt und verdeutlicht in welcher Beziehung die Daten zueinander stehen. Am Ende der Phase des Data Understanding wird versucht, anhand der gewonnen Einblicke, Erkenntnisse und Hypothesen für das weitere Vorgehen abzuleiten.

5.2.1 Aufbau eines Kalibrierprojektes

In Abbildung 21 ist der grundlegende Aufbau eines Kalibrierprojektes in AVL CRETA™ skizziert. Die wichtigsten Strukturelemente und deren Funktion sowie die dahinterliegenden Daten werden darauffolgend erläutert.

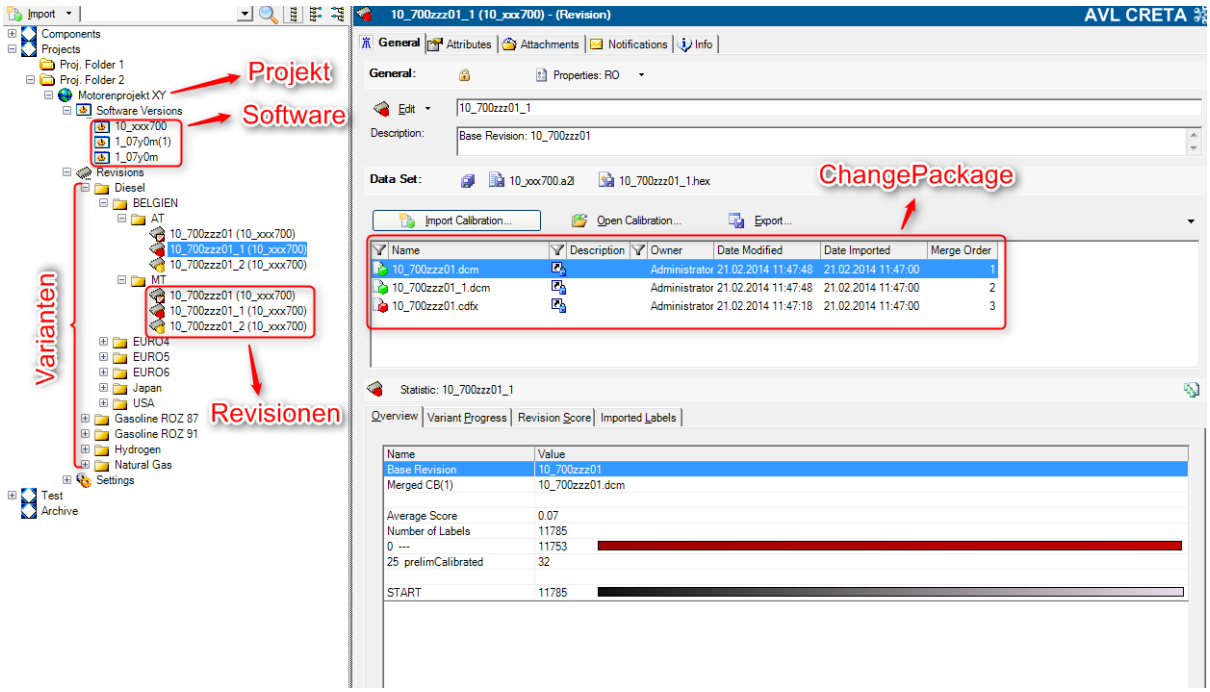


Abbildung 21: Grundlegender Aufbau eines Kalibrierprojektes.

Projekt:

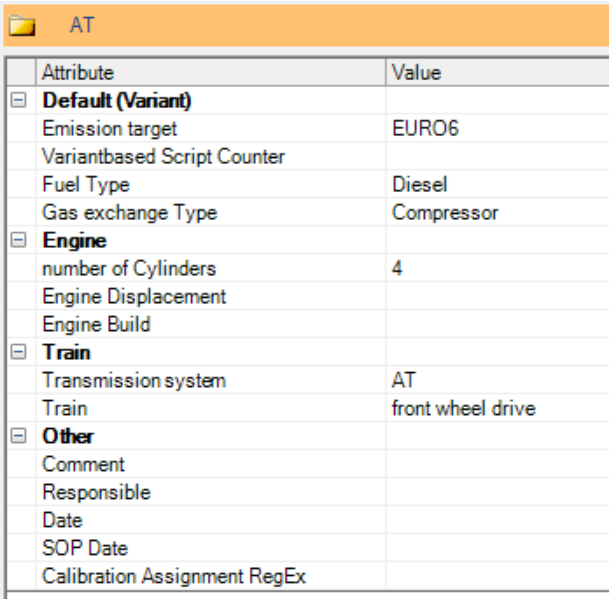
Ein Projekt ist das oberste Strukturelement in einem Kalibrierprozess und steht für ein bestimmtes Steuergerät. In der Regel handelt es sich um ein Steuergerät für einen Motor oder ein Getriebe. Es trägt daher meist einen Namen, der mit dem Motor oder Getriebe in Verbindung gebracht werden kann. Innerhalb einer solchen Projektstruktur werden sämtliche Daten verwaltet, die für den Kalibrierprozess dieses Steuergerätes benötigt werden. Auf Projektebene können allgemeine Berechtigungen vergeben werden, welche die Zugriffsrechte der unterschiedlichen Benutzer und Benutzergruppen definieren.

Varianten:

Die Varianten dienen vorrangig zur Gliederung innerhalb eines Projektes. In den Varianten werden die unterschiedlichen Kalibrierungen für das Steuergerät abgelegt, abhängig von den gewünschten Eigenschaften, die in der jeweiligen Variante erreicht werden sollen. In dem beispielhaften Kalibrierprojekt aus Abbildung 21 ist die Variantenstruktur in der ersten Ebene

nach der Art des Kraftstoffes gegliedert (Diesel, Gasoline ROZ 87/91, Hydrogen und Natural Gas), in der zweiten Ebene nach Emissionsnorm (Belgien, EURO 4-6, Japan und USA) und zuletzt nach Getriebeart (Automatic Transmission, Manual Transmission).

Abbildung 22 zeigt die Möglichkeit, auf Variantenebene selbst definierte Attribute zu vergeben, um die Eigenschaften einer Variante detaillierter zu beschreiben. Die hier vergebenen Attribute könnten theoretisch auch als Attributwerte für Data Mining Methoden von Nutzen sein, in der Praxis wird die Option der Attributvergabe für Varianten aber kaum bis gar nicht genutzt. Die verantwortlichen Applikateure wissen aus der beruflichen Erfahrung, welche Eigenschaften sich hinter den Varianten verbergen und sehen es nur als Mehraufwand in jeder Variante Attribute zu vergeben.


AT

Attribute	Value
Default (Variant)	
Emission target	EURO6
Variantbased Script Counter	
Fuel Type	Diesel
Gas exchange Type	Compressor
Engine	
number of Cylinders	4
Engine Displacement	
Engine Build	
Train	
Transmission system	AT
Train	front wheel drive
Other	
Comment	
Responsible	
Date	
SOP Date	
Calibration Assignment RegEx	

Abbildung 22: Möglichkeit der Attributvergabe auf Variantenebene.

Software:

Die Software ist als ein rein beschreibendes Format zu verstehen. Sie ist standardisiert nach ASAM MCD-2 MC[9] und enthält eine Liste aller veränderbaren Parameter (auch Labels genannt) des Steuergerätes. Außerdem beinhaltet sie Informationen über den Wertetyp, die Einheit der kalibrierbaren Parameter und die physikalischen Limits in denen ein Parameterwert verändert werden darf.

Die Software ist notwendig um eine Revision (Datei mit den Kalibrierwerten) überhaupt lesen und beschreiben zu können, da sie angibt, an welcher Speicherstelle im Revisionsfile

[9] Für genaue Informationen zu dieser Standardisierung wird auf (ASAM, n.d.) verwiesen.

die Werte eines Labels hinterlegt sind. Zusätzlich werden auf der Ebene der Software die Aufgabenbereiche der Applikateure geregelt, indem für jeden Parameter angegeben wird welche Personen für dessen Kalibrierfortschritt verantwortlich sind.

Revision:

Die Revision ist eine HEX Datei, in der letztendlich die tatsächlichen Werte der einzelnen Labels hinterlegt sind. Wie zuvor erwähnt, ist eine Revision nur zusammen mit einer Software lesbar und daher auch immer mit einer bestimmten Softwareversion explizit verknüpft. Revisionen beinhalten immer alle Parameter, die in der dazugehörigen Software hinterlegt sind. Änderungen von Werten werden nie direkt in einer Revision getätigt, sondern erfolgen über ein eigenes Dateiformat namens Change Packages, welches nachfolgend beschrieben wird.

Change Package:

Ein Change Package, auch Calibration genannt, enthält nur eine Teilmenge an Parametern und deren Änderungen. In einer Calibration werden beispielsweise die Werte hinsichtlich der Emission optimiert und anschließend zu einer Revision hinzugefügt. Unter einer Revision können viele verschiedene Change Packages mit ihren jeweiligen Änderungen eingefügt werden. Sollen diese Änderungen in eine Revision übernommen werden, muss ein sogenannter Mischprozess angestoßen werden, der zuerst die Änderungen auf Konflikte hin überprüft und anschließend eine neue Revision erzeugt. Die neue Revision beinhaltet die Änderungen aller Calibrations und die restlichen Werte der Vorgängerrevision. Basierend auf diese Vorgehensweise werden nach und nach neue Revisionen erstellt, wobei die letzte Revision unter einer Variante immer den aktuellsten Kalibrierfortschritt enthält. Abschließend sei erwähnt, dass Calibrations häufig in mehrere Revisionen zugleich eingespielt werden, da zwischen diversen Varianten Gemeinsamkeiten existieren und ein unnötiger Kalibrieraufwand vermieden werden soll.

5.2.2 Kalibrierdaten/Parameter

Die Parameter in AVL CRETA™ sind mit einem eindeutigen Namen gekennzeichnet und einem bestimmten Datentyp zugeordnet. Der Datentyp sowie die Einheit eines Labels können sich während eines Kalibrierprojektes ändern. Dies geschieht, wenn eine aktualisierte Software in das Projekt eingefügt wird. Abbildung 23 zeigt, wie Labels in AVL CRETA™ dargestellt werden.

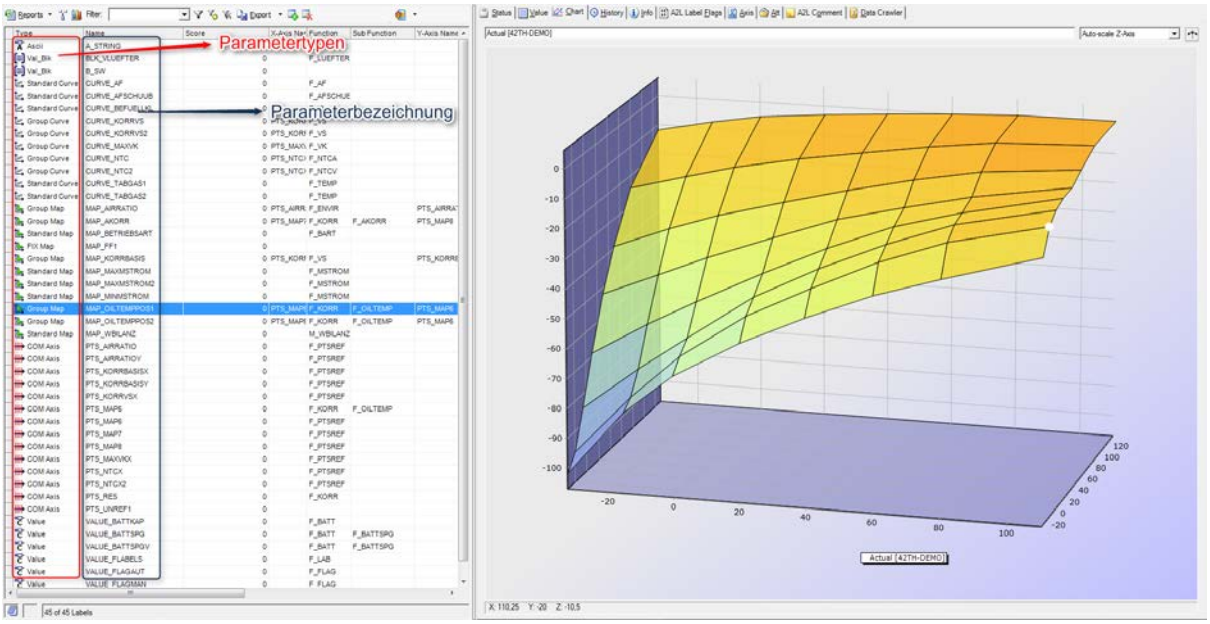


Abbildung 23: Parameter in AVL CRETA™.

Die Parameter können ein-, aber auch mehrdimensional sein. Des Weiteren sind eine numerische und eine nicht-numerische Ausprägung der Parameter möglich. Nachfolgend werden die wichtigsten Wertetypen beschrieben:

- Value: eindimensionaler Wert, welcher sowohl numerisch als auch nicht-numerisch sein kann
- Ascii: ein HEX Wert der nicht-numerisch interpretiert wird
- ValueBlock: Liste aus Einzelwerten welche numerisch sowie nicht-numerisch sind
- Axis: wie der Valueblock eine Liste von Einzelwerten, welche jedoch nur numerisch sein können
- Curve: zweidimensionaler Datentyp, der aus numerischen X- und Y-Werten zusammengesetzt ist
- Map: ein dreidimensionaler Datentyp, der aus numerischen X-, Y- und Z-Werten besteht

5.2.3 Datenmengen

Im Folgenden werden die Anzahl der unterschiedlichen Elemente und die Datenmengen eines durchschnittlichen Kalibrierprojektes skizziert, um einen Eindruck in Bezug auf die zu erwartenden Datenmengen zu bekommen:

- 85 Softwareversionen
- 7.200 Varianten

- 8.000 Revisionen
- 29.000 Calibrations

Alle Dateien eines Kalibrierprojektes verbrauchen in Summe in etwa 150 Gigabyte. Die Softwarefiles und Revisionen enthalten ungefähr 45.000 unterschiedliche Parameter. Die Dateigröße einer Software liegt bei ca. 12 Megabyte und jene einer Revision bei ca. 15 Megabyte. Calibrations enthalten im Schnitt nur 50 geänderte Parameter und sind ca. 15 Kilobyte groß.

Sämtliche Daten werden zentral in einem Datenbankmanagementsystem verwaltet. Die unterschiedlichen Dateiformate sind in der Datenbank im ZIP-Format abgelegt. Fehlende Werte sind auszuschließen, da ein solcher Umstand Revisionen unlesbar machen würde. Attributwerte für die Daten könnten theoretisch auf Variantenebene vergeben werden, was jedoch in der Praxis nicht genutzt wird. Das einzige Attribut, welches den Datenstand sehr grob beschreibt, ist der Variantenpfad. Die Variantenstruktur kann zwischen den Projekten Ähnlichkeiten haben, ist in der Regel aber unterschiedlich, was ein projektübergreifendes Lernen unmöglich macht.

5.2.4 Fazit

Es kann zusammenfassend festgestellt werden, dass eine Ansammlung unterschiedlicher Parameter ohne ausreichende Beschreibung vorhanden ist. Aufgrund der fehlenden Beschreibung kann bereits zu diesem Zeitpunkt angenommen werden, dass Data Mining Methoden für Prognoseprobleme mit diesen Daten nicht anwendbar sind. Diese Methoden würden als Eingabeformat Instanzen mit Attributen (Eigenschaften die den Klassenwert beeinflussen bzw. beschreiben) und einem Klassenwert als Ergebnis erwarten. Außerdem kann, aus dem Umstand wie die Daten gespeichert sind, vermutet werden, dass eine Datenaufbereitung durchgeführt werden muss, bevor eine Data Mining Methode anwendbar ist. Dieser Aufgabe wird sich das nächste Kapitel annehmen.

5.3 Data Preparation

Die zyklische Eigenschaft des CRIPS-DM Prozesses würde es vorsehen, dass die Entscheidung über die verwendeten Methoden erst im Kapitel des Modeling erfolgt und anschließend ein Rückschritt zur Datenaufbereitung durchgeführt wird. Dies empfiehlt sich in der Praxis, ist aber in Bezug auf die Strukturierung einer schriftlichen Arbeit nicht vorteilhaft. Daher wird in diesem Kapitel vorgegriffen, welche Data Mining Methoden gewählt werden.

Dies hat unter anderem nämlich auch einen entscheidenden Einfluss auf die Datenaufbereitung.

Prognoseprobleme konnten aufgrund der fehlenden Attributwerte bereits in der Phase des Data Understanding ausgeschlossen werden. Die Erkenntnisse aus dieser Phase und deren Auswirkung auf die Data Mining Ziele wurden mit den Domänenexperten diskutiert. Aus dieser Diskussion ging hervor, dass sich vermutlich Muster aus der Art und Weise wie Calibrations abgegeben werden ableiten lassen. Änderungen der Werte werden in Form von Calibrations meist in mehrere Varianten zugleich abgegeben. Zwischen den Varianten gibt es daher Zusammenhänge, welche sich aus dem Abgabeverhalten erkennen lassen sollten.

Diese Zusammenhänge weisen große Ähnlichkeit mit einer klassischen Warenkorbanalyse auf. Bei einer Warenkorbanalyse enthält eine Transaktion alle Artikel, die ein Kunde gemeinsam gekauft hat und aus der Sammlung von Transaktionen kann die Information gewonnen werden welche Artikel Abhängigkeiten aufweisen und bevorzugt zusammen gekauft werden. Mit demselben Ansatz können auch Abhängigkeiten zwischen den Varianten aufgefunden werden. Eine Transaktion wäre somit die Menge der Varianten in denen dieselbe Calibration abgegeben wurde. Dieser Ansatz kann mit den vorhandenen Informationen durchgeführt werden und wurde auch von den Domänenexperten als durchaus interessant eingestuft. Aus diesem Grund wurde die Assoziationsanalyse als geeignete Methode für das weitere Vorgehen ausgewählt.

Ein möglicher Anwendungsfall aus dem Kapitel des Business Understanding war, dass Falschabgaben von Kalibrierdaten erkannt werden sollten. Laut Aussagen von Domänenexperten ist ein typischer Fehler, dass Kalibrierdaten versehentlich nicht in allen notwendigen Variantenzweigen abgegeben werden. Es wäre das Ziel, dass dieses Fehlverhalten erkannt und weiters verhindert wird. Die folgenden Schritte dieser Arbeit legen den Fokus in die Umsetzung dieser Anforderung.

Aufgrund der gewählten Methode der Assoziationsanalyse müssen die vorhandenen Daten entsprechend aufbereitet werden. Laut (IBM-Corporation, 2012) gibt es zwei gängige Formate, aus denen Assoziationsmodelle erstellt werden. Diese beiden Formate werden nachfolgend skizziert.

In Tabelle 6 ist das Transaktionsformat als Datenbasis dargestellt. Dieses Format wird auch als Kassenrollen-Format bezeichnet und es enthält für jeden Datensatz eine eigene

Transaktion in der Tabelle. In diesem Beispiel wurde die Calibration_3 in drei unterschiedliche Varianten abgegeben woraus auch drei Einträge in der Tabelle folgen. (IBM-Corporation, 2012)

Calibration	Variante
Calibration_1	Variante_4
Calibration_2	Variante_1
Calibration_2	Variante_2
Calibration_3	Variante_1
Calibration_3	Variante_2
Calibration_3	Variante_4
Calibration_4	Variante_3
Calibration_4	Variante_4

Tabelle 6: Ausgangsdaten im Transaktionsformat.

In Tabelle 7 ist das Tabellendaten-Format dargestellt. Weitere gebräuchliche Bezeichnungen für dieses Format sind Warenkorbdaten und Wahrheitstabellendaten. In diesem Format wird über ein Flag gekennzeichnet, ob Elemente in einer Transaktion vorhanden sind. Dieses Format zeichnet sich vor allen durch eine bessere Lesbarkeit aus. (IBM-Corporation, 2012)

Calibration	Variante_1	Variante_2	Variante_3	Variante_4
Calibration_1	0	0	0	1
Calibration_2	1	1	0	0
Calibration_3	1	1	0	1
Calibration_4	0	0	1	1

Tabelle 7: Ausgangsdaten im Tabellendatenformat.

Die Daten für den zu implementierenden Prototypen werden in das Transaktionsformat übergeführt, da die Speicherung in diesem Format kompakter ist und bei einer hohen Variantenvielfalt das festgesetzte Limit der maximalen Spaltenanzahl mit dem Tabellenformat überschritten werden könnte.

Im Folgenden wird beschrieben, wie die vorhanden Daten in das Transaktionsformat übergeführt werden. Abbildung 24 zeigt die Abgabe der Calibrations in mehrere Varianten gleichzeitig. Nach der Abgabe wird in der Datenbank für jede Variante eine eigene

Calibration angelegt. In der Datenbank gibt es jedoch keine direkte Information, dass diese Calibrations identisch sind, da sie aus einer gemeinsamen Datei entstanden sind.

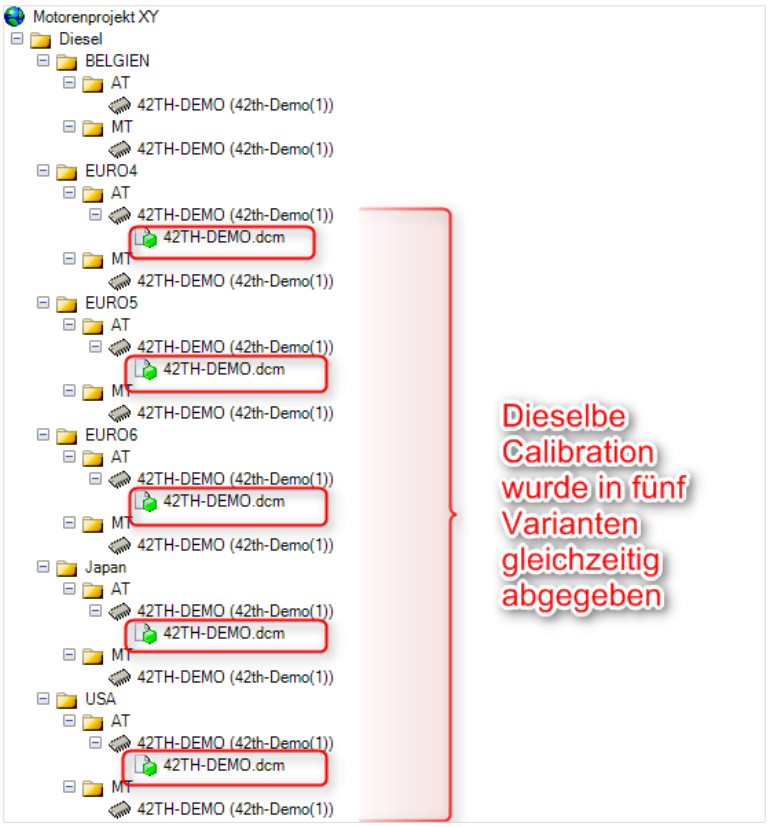


Abbildung 24: Abgabe derselben Calibration in mehrere Varianten.

Aus diesem Grund muss im ersten Schritt die Gruppierung dieser Calibrations erfolgen. In der Datenbank ist ein CRC32 Prüfwert für jede Datei abgelegt, wodurch sich die zusammengehörenden Dateien recht zuverlässig gruppieren lassen. Zusätzlich werden noch der Zeitpunkt des Importes und die Anzahl der Labels herangezogen, um die Wahrscheinlichkeit einer korrekten Gruppierung zu erhöhen.

Basierend auf dieser Gruppierung kann anschließend eine neue Tabelle im Transaktionsformat aufgebaut werden, in welcher die Information vorhanden ist, welche Calibration zugleich in mehrere Varianten abgegeben wurde. Diese Datenbasis ist nun auch dafür geeignet, die Data Mining Methode der Assoziationsanalyse auf sie anzuwenden. Außerdem wurden alle Elemente der Transaktionstabelle in einen Int32 Wert überführt, um einerseits Datenbankspeicher zu sparen und um andererseits die Transaktionstabelle später auch im Arbeitsspeicher halten zu können.

5.4 Modeling

In diesem Kapitel wird beschrieben, wie die in Kapitel 4 vorgestellten Algorithmen im Prototyp verwendet werden. Zuerst werden die Eigenschaften des Apriori- und FP-growth Algorithmus noch einmal einander gegenübergestellt, dann wird über diesen Vergleich begründet, welcher dieser beiden Algorithmen zur Assoziationsanalyse im Prototyp implementiert wurde.

Außerdem wird auf die Größe der Ergebnismenge und die damit verbundenen Probleme eingegangen. Es werden die wichtigsten Möglichkeiten aufgezeigt wie diese Ergebnismenge weiterverarbeitet werden kann und es wird dargelegt, welche Methoden für den Prototyp verwendet wurden. Abschließend erfolgt eine Zusammenfassung der verwendeten Methoden in Form einer grafischen Darstellung.

Erwähnt sei in diesem Zusammenhang auch noch das Open Source Tool WEKA[10] der Neuseeländischen Universität Waikato. In diesem Tool sind eine Fülle von Data Mining Algorithmen implementiert, wodurch recht schnell erste Versuche mit unterschiedlichen Algorithmen gemacht werden können. Die beiden oben genannten Algorithmen sind ebenfalls in WEKA implementiert.

5.4.1 Auswahl Algorithmus

Der FP-growth Algorithmus liefert, wie der Apriori, alle *Frequent-Patterns* aus einer Datenbasis indem Schwellwerte für den *Support* und für die *Confidence* festgelegt werden. Die beiden Algorithmen unterscheiden sich lediglich in der Arbeitsweise, aber die Ergebnismenge der beiden Algorithmen ist identisch.

Der FP-growth – verzichtet im Gegensatz zum Apriori – auf die Generierung von *Candidatesets* und ermittelt die *Frequent Patterns* stattdessen über die kompakte Datenstruktur des FP-Tree. Aufgrund dieser kompakten Datenstruktur sind beim FP-growth die Anzahl der Datenbankzugriffe niedriger, der Speicherverbrauch geringer und die Laufzeit kürzer. Aus diesem Grund wurde der FP-growth Algorithmus auch im Prototyp implementiert. Die folgende Tabelle 8 stellt die Eigenschaften der beiden Algorithmen vergleichend gegenüber.

[10] http://www.cs.waikato.ac.nz/ml/weka/

Vergleich	Apriori	FP-growth
Ergebnis	Alle *Frequent-Patterns* die einen Minimum *Support* erfüllen.	Alle *Frequent-Patterns* die einen Minimum *Support* erfüllen.
Arbeitsweise	Über den *Join*- und *Prune Step* werden mögliche Candidatesets gebildet. Aus diesen werden dann die *Frequent-Patterns* ermittelt.	Erstellen des FP-Tree und anschließende Bestimmung der *Frequent-Patterns* über die *Conditional Pattern Base* und den *Conditional FP-Tree*.
Speicherverbrauch	Wird erhöht durch die benötigte Generierung der Candidatesets.	Durch die kompakte Datenstruktur des FP-Tree und den Wegfall der Generierung von Candidatesets ist der Speicherverbrauch geringer.
# Datenbankscans	Abhängig von der Apriori Version. Im schlechtesten Fall bei jedem k-ten Durchlauf. In der AprioriTid Version bestenfalls einmal, jedoch führt dies bei einer größeren Datenbasis zu hohen Input/Output-Kosten.	Kommt mit exakt zwei Scans der Datenbank aus.
Performance	Langsamer, da viel Rechenzeit durch die Generierung der Candidatesets beansprucht wird.	Erspart sich die Generierung der Candidatesets, was den Rechenaufwand vermindert.

Tabelle 8: Vergleich Apriori- und FP-growth-Algorithmus.

5.4.2 Größe der Ergebnismenge

Für den Prototyp wurden die gewonnen Regeln noch über die in Kapitel 4.1.7 beschriebenen Interessantheitsmaße reduziert. Dennoch ist in Bezug auf die Performanz und Rechnerkapazität zu beachten, dass die Ergebnismenge schnell explodieren kann. Die aus (Hippner, Küsters, Meyer, & Wilde, 2001) entnommene Tabelle 9 stellt den exponentiellen Anstieg der Kombinationsmöglichkeiten für eine Grundmenge von 100 *Item*s mit *Itemsets* der Größe k dar.

k	Kombinationsmöglichkeiten
1	100
2	4.950
3	161.700
4	3.921.225
5	75.287.520
6	1.192.052.400
7	16.007.560.800
8	186.087.894.300

Tabelle 9: Kombinationsmöglichkeiten für Itemsets der Größe k mit 100 Elementen.

Ergänzend sei erwähnt, dass die Anzahl der Kombinationen eine *Kombination ohne Wiederholungen* ist, die folgendermaßen errechnet wird:

$$\frac{\boldsymbol{n}!}{(\boldsymbol{n}-\boldsymbol{k})!\,\boldsymbol{k}!}$$

Die entsprechende Wahl der *Support*- und der *Confidence*-Schwellwerte begrenzen dieses Wachstum und sollten mit entsprechendem Bedacht gewählt werden. In der Datenbasis wurden unterschiedliche Projekte hinsichtlich des Kombinationsproblems untersucht. Ein starker Anstieg des Rechenaufwandes konnte erst mit einem *Supportwert* < 5 Prozent festgestellt werden.

Die Anforderungen an das Ergebnis eines Data Mining Prozesses besagen – laut Definition in Kapitel 3.2 – dass die gewonnen Informationen neu, potentiell nützlich und letztendlich verständlich sein sollen. Diese Anforderungen werden allein durch die Generierung von Assoziationsregeln nur teilweise erfüllt. Der nachfolgende Abschnitt beschäftigt sich mit der Weiterverarbeitung der bisher gewonnen Informationen, um den in Kapitel 5.3 definierten

Anwendungsfall – Falschabgaben von Kalibrierdaten sollen erkannt werden – erfüllen zu können.

5.4.3 Probleme der Auswertung von Assoziationsregeln

Ein bekanntes Problem der Assoziationsanalyse ist, dass diese als Ergebnis eine für den Menschen nicht mehr überblickbare Menge an Assoziationsregeln liefert. In (Hippner, Küsters, Meyer, & Wilde, 2001, S. 444) werden folgende drei Methoden aufgezeigt, um dieses Problem zu reduzieren:

- Anhand diverser Eigenschaften können die Regeln sortiert werden. Mögliche Merkmale wären die Antezedens oder Konsequenz der Regel, *Support*, *Confidence*, Interessantheitsmaß, *Lift*, *Conviction* etc.
- Implementierung einer Filterfunktionalität, die nur Regeln anhand einer Bedingung abfragt. Beispielsweise nur Regeln, die ein definiertes *Item* enthalten bzw. nicht enthalten, oder alle Regeln mit *Item1* ohne *Item2* und einer *Confidence* > 80 Prozent.
- Grafische Darstellung[11] der Gesamtmenge an Assoziationsregeln, um eine Überblick zu bekommen, damit der Anwender Muster und Zusammenhänge erkennen kann. Dieser Ansatz stößt jedoch aufgrund der meist vorhandenen Fülle und Komplexität der Regeln bald an seine Grenzen.

Diese drei Methoden helfen nur eingeschränkt aussagekräftige Informationen aus der entstandenen Regelmenge herauszufiltern. Ein weiterer Ansatz wäre es, in den Transaktionsdaten gezielt nach jenen Transaktionen zu suchen, die von anderen abweichen. Das folgende Kapitel beschreibt wie diese Methoden im Prototyp verwendet wurden.

5.4.4 Anwendung der Ausreißererkennung im Prototyp

Die beiden in Kapitel 4.2 beschriebenen Ausreißermaße von (He, Huang, & Deng, 2005) und (Narita & Kitagawa, 2008) wurden im Prototyp für jede Transaktion berechnet. Letztendlich wurden alle Transaktionen für den Anwender in tabellarischer Form in einer grafischen Benutzeroberfläche aufbereitet. Über Filterfunktionen und der Möglichkeit zu Sortierung wurden ihm weitere Hilfsmittel zur Verfügung gestellt, um die Anzahl der abweichenden Transaktionen soweit einzuschränken, dass diese überschaubar sind und gegebenenfalls einer Nachprüfung zugeführt werden können (z.B. sortieren nach Ausreißermaß und Ausfiltern von

[11] In (Vodenicharov, 2007) werden unterschiedliche Visualisierungsmöglichkeiten von Assoziationsregeln beleuchtet.

Abgaben mit weniger als zwei Varianten). Einen Überblick zu den wichtigsten Arbeitsschritten und angewandten Methoden liefert das anschließende Kapitel.

5.4.5 Zusammenfassung Modell

Eine Zusammenfassung der Arbeitsschritte zur Erkennung von Falschabgeben gibt die Abbildung 25.

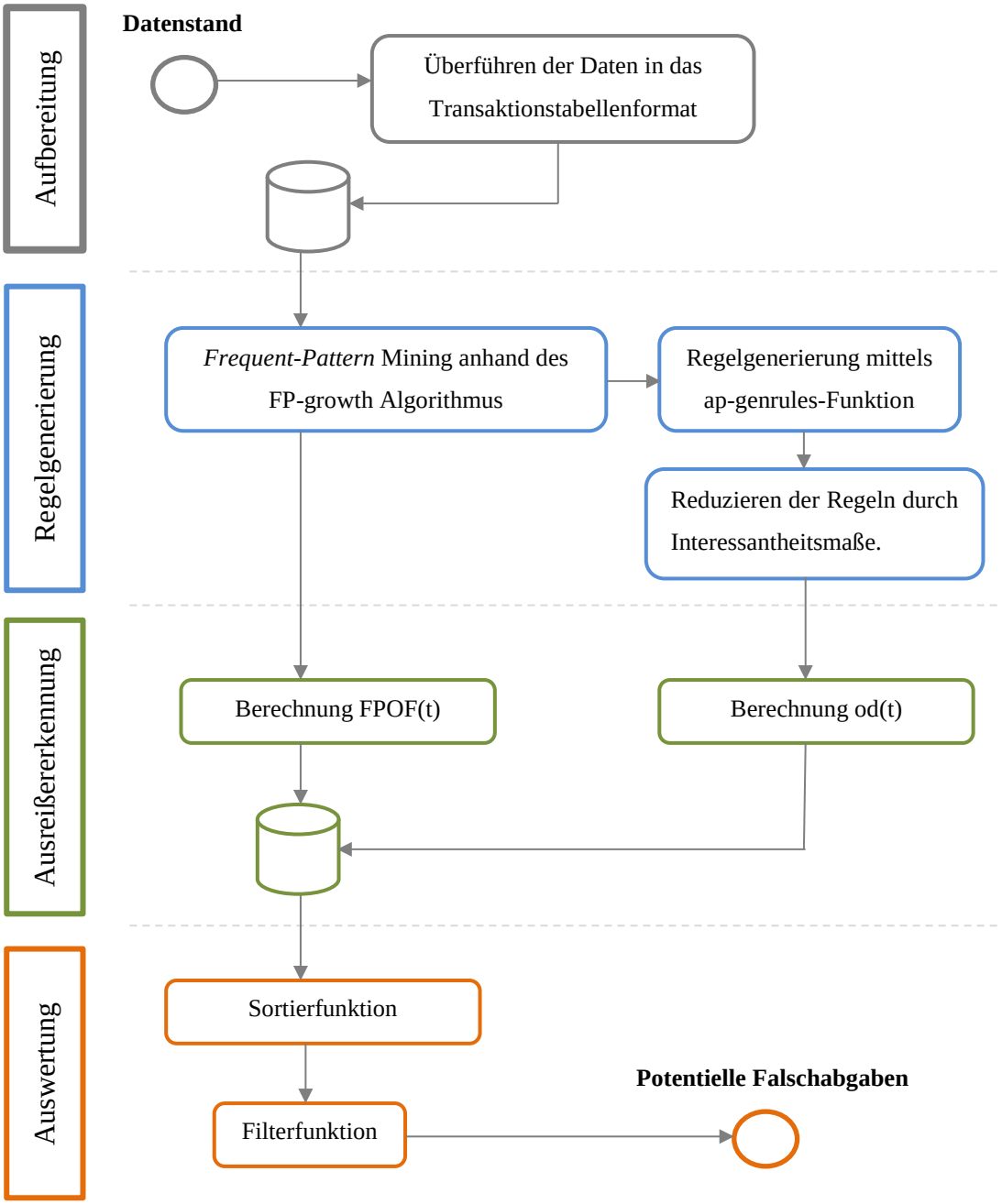


Abbildung 25: Übersicht angewandter Methoden.

5.5 Evaluation

Mit den zuvor gewählten Methoden wurde ein Prototyp entwickelt, der die Abgaben von Calibrations anhand der in Kapitel 4.2 vorgestellten Maße zur Ausreißererkennung bewertet. In diesem Kapitel werden die Ergebnisse des Prototyps evaluiert. Der Prototyp wurde im ersten Schritt mit einem historischen Projekt getestet, in welchem es zu einem Abgabefehler gekommen ist. Danach folgt ein Test mit einem zweiten Projekt, in dem mehrere Falschabgaben absichtlich produziert wurden.

5.5.1 Evaluierung mit echter Falschabgabe

Dieser Test wurde mit einem historischen Projekt durchgeführt, in welchem die Abgabe in einem ganzen Variantenzweig vergessen wurde. Es wurde eine Calibration in nur 11 Varianten abgegeben statt in vorgesehenen 57 Varianten.

Um die Falschabgaben zu finden, wurde für alle Transaktionen der FPOF(t) und der od(t) bestimmt. Die Transaktionen wurden anschließend jeweils nach diesen Ausreißermaßen sortiert. Die *Frequent-Patterns* wurden mit einem *Support* von 10 Prozent bestimmt und zur Regelgenerierung wurde für die *Confidence* ein Schwellwert von 85 Prozent festgesetzt und zur weiteren Filterung ein *Lift* > 1.

Mit dem od(t)-Wert konnte die falsche Abgabe bereits an der siebten Position von insgesamt 1701 Transaktionen gefunden werden, was ein sehr zufriedenstellendes Ergebnis war. Viele der anderen Transaktionen, welche ebenfalls einen hohen Ausreißwert aufwiesen, wurden von Hand nachkontrolliert und haben tatsächlich Anomalien aufgewiesen. Die Ursache für den erhöhten Wert war damit begründet, dass diese Transaktionen in der Vergangenheit zum Teil wirklich eine Falschabgabe waren. Dieser Fehler wurde noch im Projektverlauf rechtzeitig erkannt und mit einer späteren Abgabe wurden die fehlenden Werte wieder in die entsprechenden Varianten übernommen. Ein anderer Teil der erkannten Falschabgaben wurde zuvor nur eine einzelne Variante abgegeben und erst nach einem Test am Prüfstand in die restlichen Varianten übernommen.

Generell kann über das Ergebnis mit diesem Maß gesagt werden, dass auch die Anzahl der Transaktionen mit einem erhöhten Ausreißerwert überschaubar sind und bei einer Überprüfung durch den Applikateur meist eine schlüssige Ursache für den erhöhten Wert gefunden werden konnte. Die Verteilung der Werte ist in Abbildung 26 dargestellt.

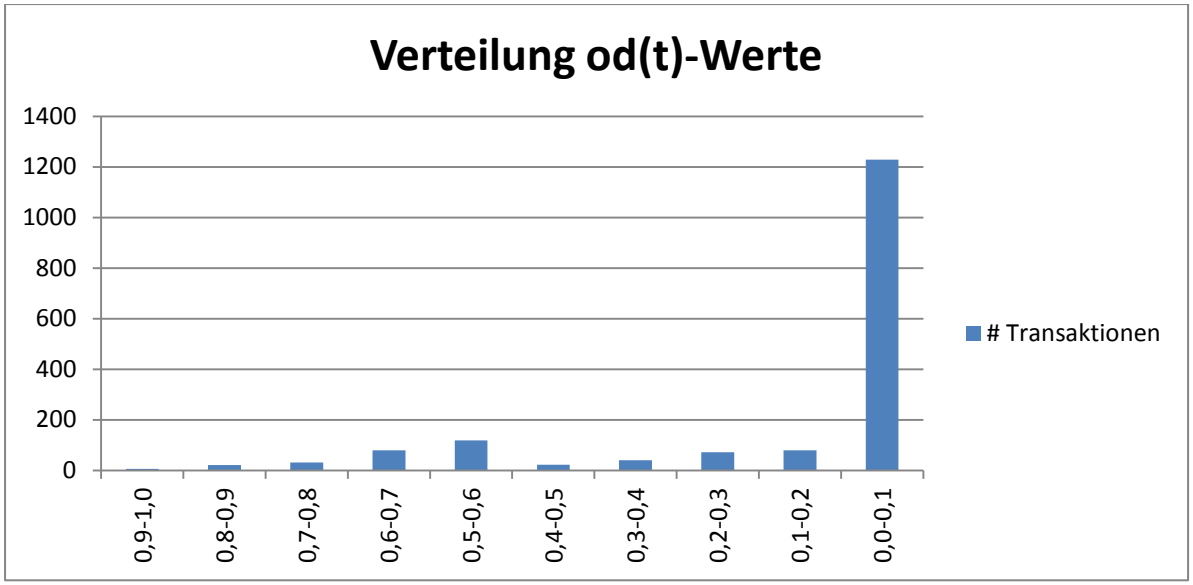


Abbildung 26: Verteilung des od(t)-Maßes.

Der FPOF(t) lieferte mit diesem Projekt kein annehmbares Ergebnis. Die Falschabgabe wurde überhaupt nicht erkannt und dieses Maß stufte die Falschabgabe als unauffällig ein. Der FPOF(t) stuft die Transaktionen mit den kleinsten Werten als Ausreißer ein. Insgesamt hatten 612 Transaktionen den Wert 0, was den maximalen Ausreißer darstellt. Diese Anzahl an Transaktionen war auch zu hoch für eine manuelle Kontrolle der einzelnen Transaktionen. Der FPOF(t) wurde nochmals mit einem niedrigeren *Support* errechnet, was das Ergebnis nur leicht verbesserte. Die Falschabgabe wurde trotzdem nicht erkannt. Bei der Überprüfung einzelner Transaktionen, die einen Ausreißer darstellen, konnte keine nachvollziehbare Ursache dafür gefunden werden.

Die Ursache der schlechten Ergebnisse des FPOF(t) für die vorhandene Problemstellung liegt an der Art, wie dieser Ausreißer bewertet. Vereinfacht kann gesagt werden, dass diese Methode eine Transaktion mit wenigen *Frequent-Patterns* als Ausreißer einstuft. In der vorhandenen Datenbasis gibt es viele Transaktionen, die wenige *Frequent-Patterns* enthalten. Dies sind Varianten, für welche kein Abgabemuster festgestellt werden kann, was jedoch nicht auf eine Falschabgabe schließen lässt. Dieses Maß bewertet viel mehr, wie stark sich eine Transaktion vom gesamten Rest der Transaktionen unterscheidet und liefert für die gegebene Problemstellung nur die Aussage, dass sich eine Abgabe nicht so oft wiederholt wie andere Abgaben. Da nur ein Teil der Abgaben einem Muster folgt und dieser Umstand normal für einen Kalibrierprozess ist, kann diese Methode für die Ausreißererkennung in diesem Anwendungsfall auch keine guten Ergebnisse liefern. Aus diesem Grund wird das FPOF(t) in der weiteren Evaluierung auch nicht mehr herangezogen.

5.5.2 Evaluierung mit künstlichen Falschabgaben

Um die Stärken und Schwächen des Prototyps festzustellen, wurden unterschiedliche Arten von Falschabgaben, die in Tabelle 10 aufgelistet sind, absichtlich durchgeführt. Anschließend wurde überprüft wie der Prototyp darauf reagiert.

ID	# abgegebene Varianten	# erforderliche Varianten	Beschreibung
T1	1	7	Varianten, die in Assoziationsregeln enthalten sind
T2	6	7	Varianten, die in Assoziationsregeln enthalten sind
T3	15	33	Varianten, die in Assoziationsregeln enthalten sind
T4	22	7	Varianten, die in Assoziationsregeln enthalten sind
T5	2	12	Varianten, die NICHT in Assoziationsregeln enthalten sind
T6	6	7	Varianten, die NICHT in Assoziationsregeln enthalten sind

Tabelle 10: Übersicht künstlicher Falschabgaben.

Aufgrund der Testergebnisse konnten folgende Eigenschaften für die benutzte Methode festgestellt werden:

- Für Varianten, die nicht in Assoziationsregeln vorhanden sind, werden keine Ausreißer erkannt, was auch aufgrund der Art, wie das Maß od(t) berechnet wird, nachvollziehbar ist. Eine Möglichkeit, dieses Verhalten zu reduzieren wäre den *Supportwert* zu senken, um mehr Regeln zu erhalten, was jedoch wiederum den Rechenaufwand erhöht.
- Erfolgt eine Abgabe in zu viele Varianten bzw. falsche Varianten, schlägt diese Methode auch nicht an. Dieser Anwendungsfall war auch nicht im Fokus der Problemstellung, jedoch wird eine mögliche Lösung dafür im Kapitel 7 angesprochen.

- Die besten Ergebnisse liefert die Methode, wenn eine hohe Anzahl an Varianten bei der Abgabe übersehen wurde. Je niedriger die Anzahl der vergessenen Varianten im Verhältnis zu den abgegebenen Varianten ist, desto kleiner ist auch der Ausreißerwert.

6 Prototyp

In diesem Kapitel erfolgt die Beschreibung des implementierten Prototyps. Auf Wunsch der AVL GmbH wurden in den Screenshots die Strukturelemente unkenntlich gemacht, da Bezeichnungen von Kundenprojekten enthalten sind, die der Geheimhaltung unterliegen.

6.1 Hauptdialog

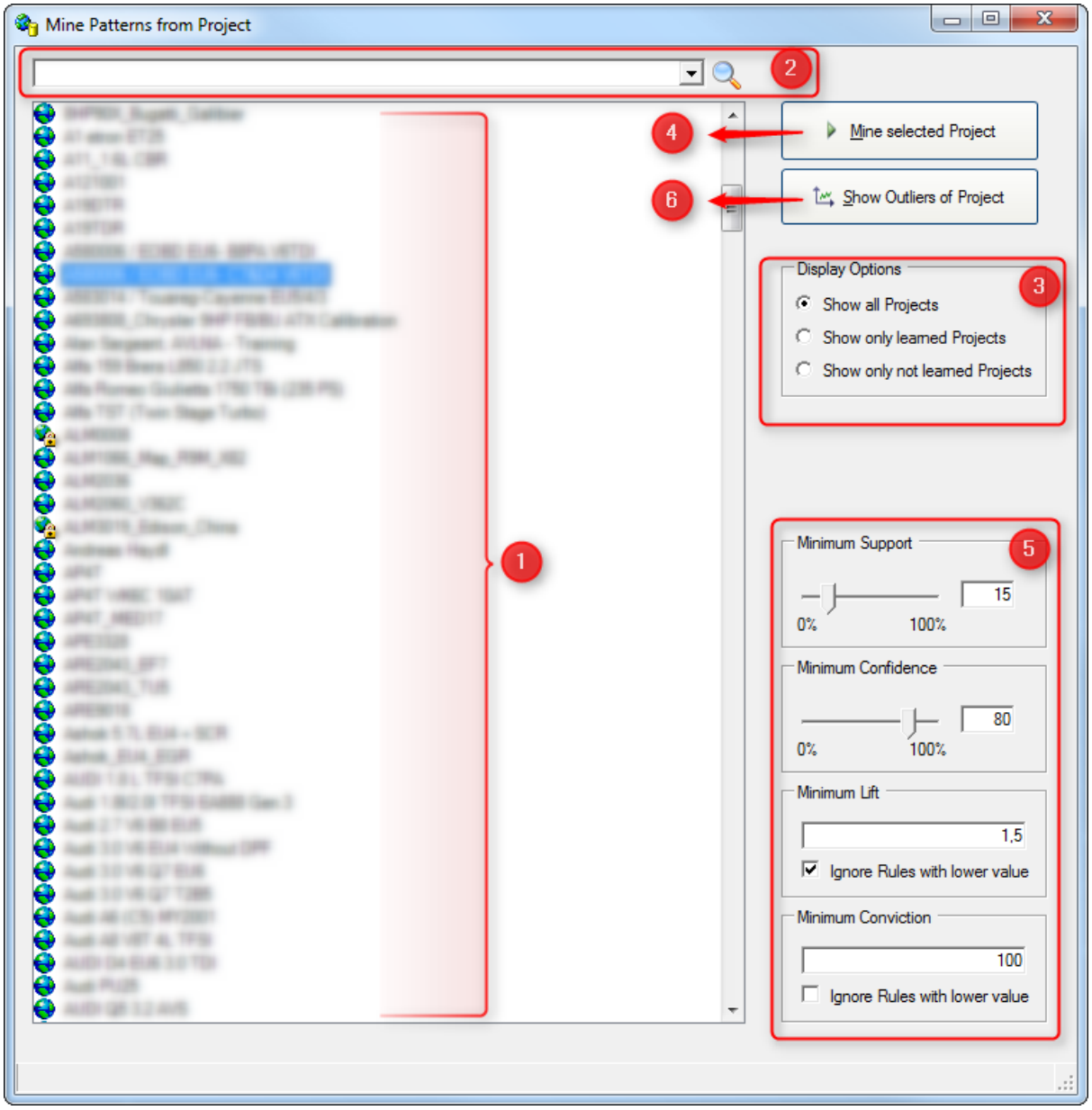


Abbildung 27: Hauptdialog.

Die Abbildung 27 illustriert den Hauptdialog mit seinen wichtigsten Funktionen. Diese sind in der Abbildung durchnummeriert und werden nachfolgend entsprechend der Nummer kurz beschrieben:

1. Auflistung aller vorhandenen Projekte, für welche Abgabemuster gelernt und anschließend mögliche Ausreißer angezeigt werden können
2. Eine Möglichkeit zur Suche nach einem bestimmten Projekt
3. Filtermöglichkeit der aufgelisteten Projekte, um trotz der großen Anzahl der Elemente eine bessere Übersicht zu haben
4. Berechnung der *Frequent-Patterns*, der Assoziationsregeln und Ausreißermaße für das jeweils selektierte Projekt
5. Einstellbare Parameter wie *Support*, *Confidence*, *Lift* und *Conviction*, nach welchen das Projekt gelernt wird
6. Öffnen des nachfolgenden Dialoges, der die Abgaben mit den Ausreißermaßen darstellt

6.2 Übersichtsdialog mit den Abgaben in einem Projekt

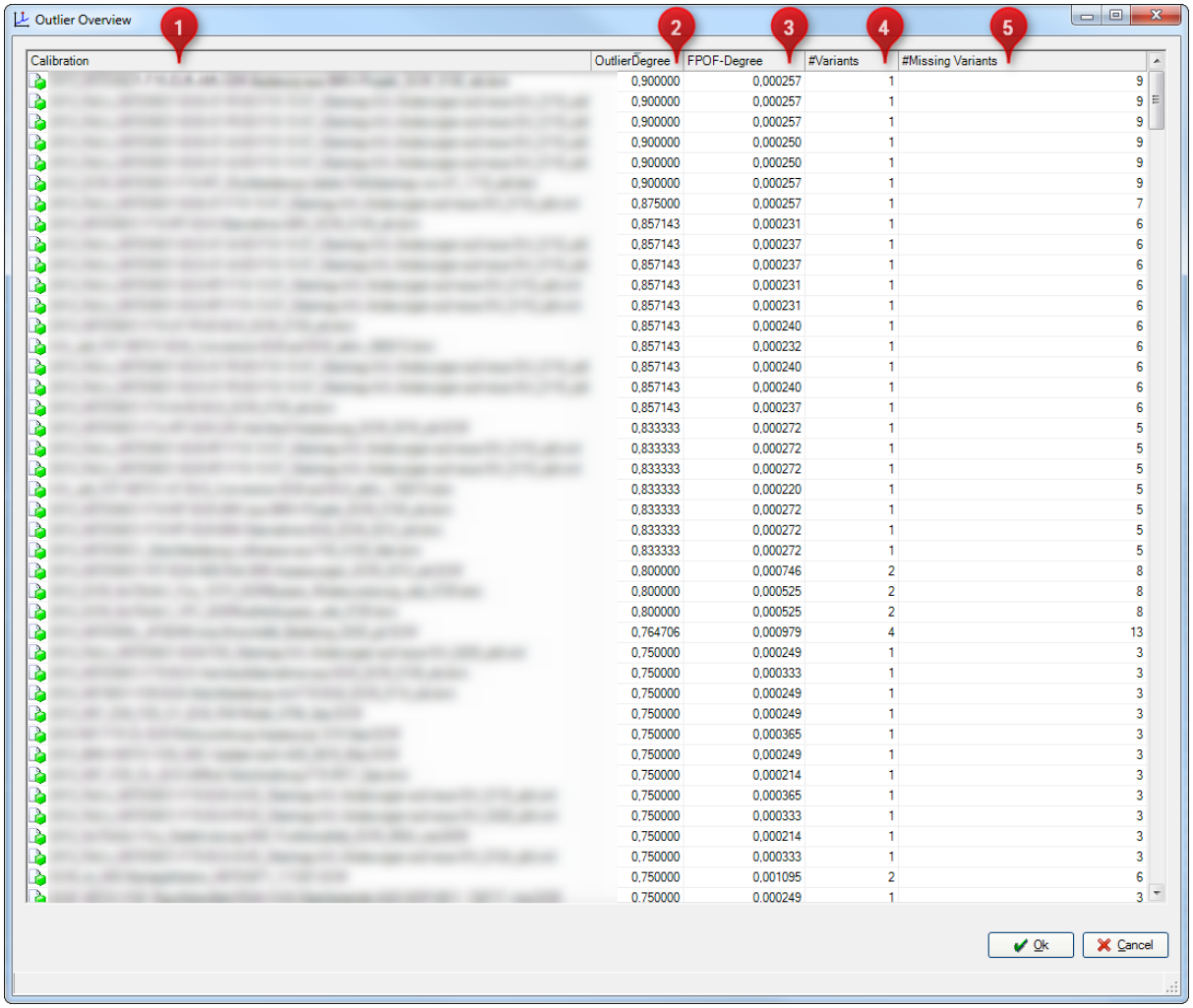


Abbildung 28: Übersichtsdialog zu abgegebenen Calibrations.

Im Dialog aus Abbildung 28 wird die Abgabe der Calibrations dargestellt. Eine Zeile in der Grid-Ansicht steht für eine Abgabe derselben Calibration in eine oder mehrere Varianten. Im Folgenden werden die einzelnen Spalten erklärt:

1. Eine abgegebene Gruppe an gleichen Calibrations
2. Das in Kapitel 4.2.2 beschriebene Ausreißermaß für diese Transaktion
3. Das in Kapitel 4.2.1 beschriebene Ausreißermaß für diese Transaktion
4. Anzahl an Varianten, in denen diese Calibration abgegeben wurde
5. Anzahl der Varianten, in welche nicht abgegeben wurde, die jedoch nach der in Kapitel 4.2.2 beschriebenen *Associative Closure* in der aktuellen Abgabe fehlen

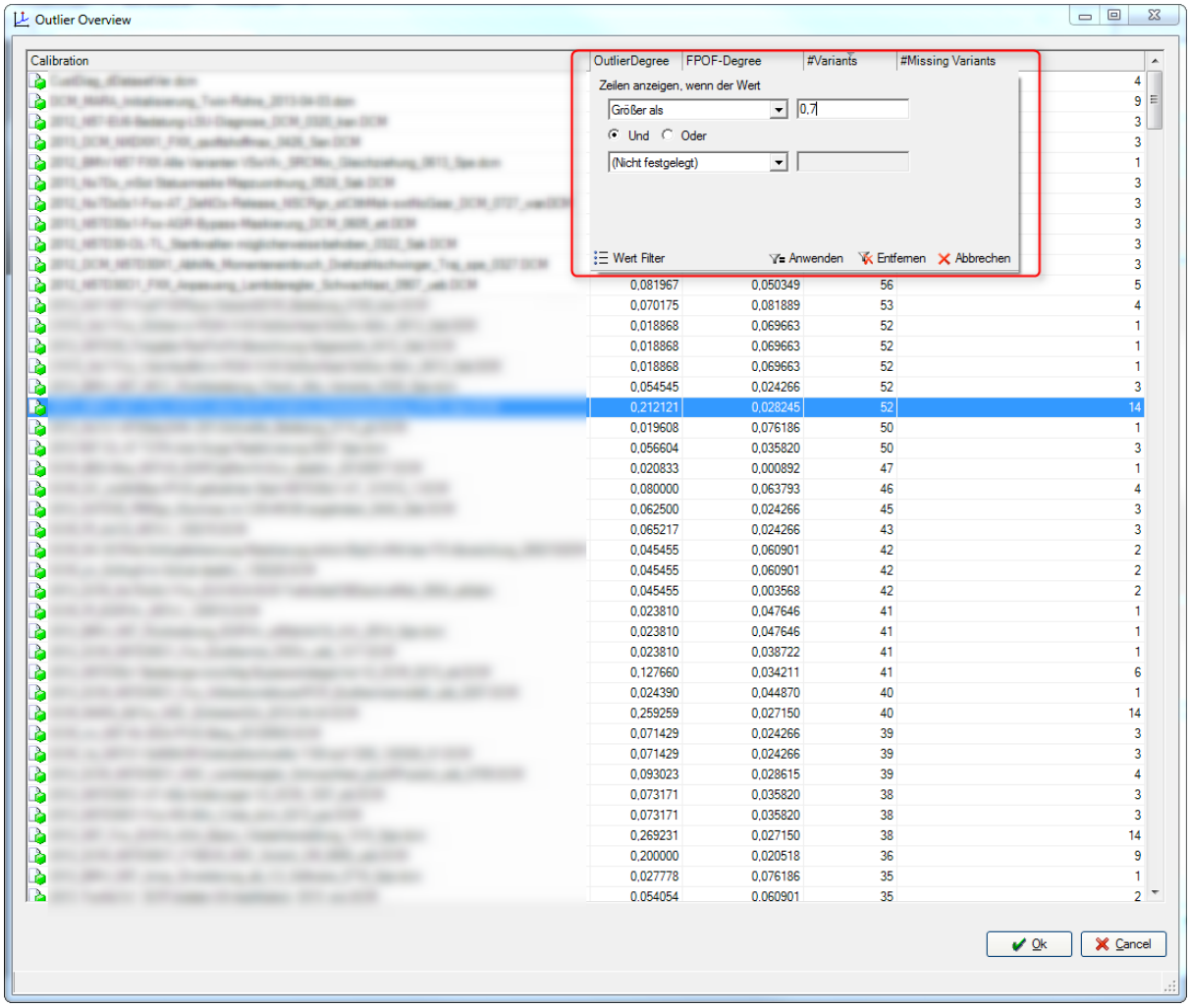


Abbildung 29: Filtermöglichkeit im Ausreißerdialog

Abbildung 29 zeigt außerdem noch die Möglichkeit Ausreißer durch Filteroptionen und Sortiereigenschaften schneller zu finden. Durch einen Doppelklick einer Zeile im Grid wird der nächste Dialog geöffnet.

6.3 Dialog mit einzelner Abgabe

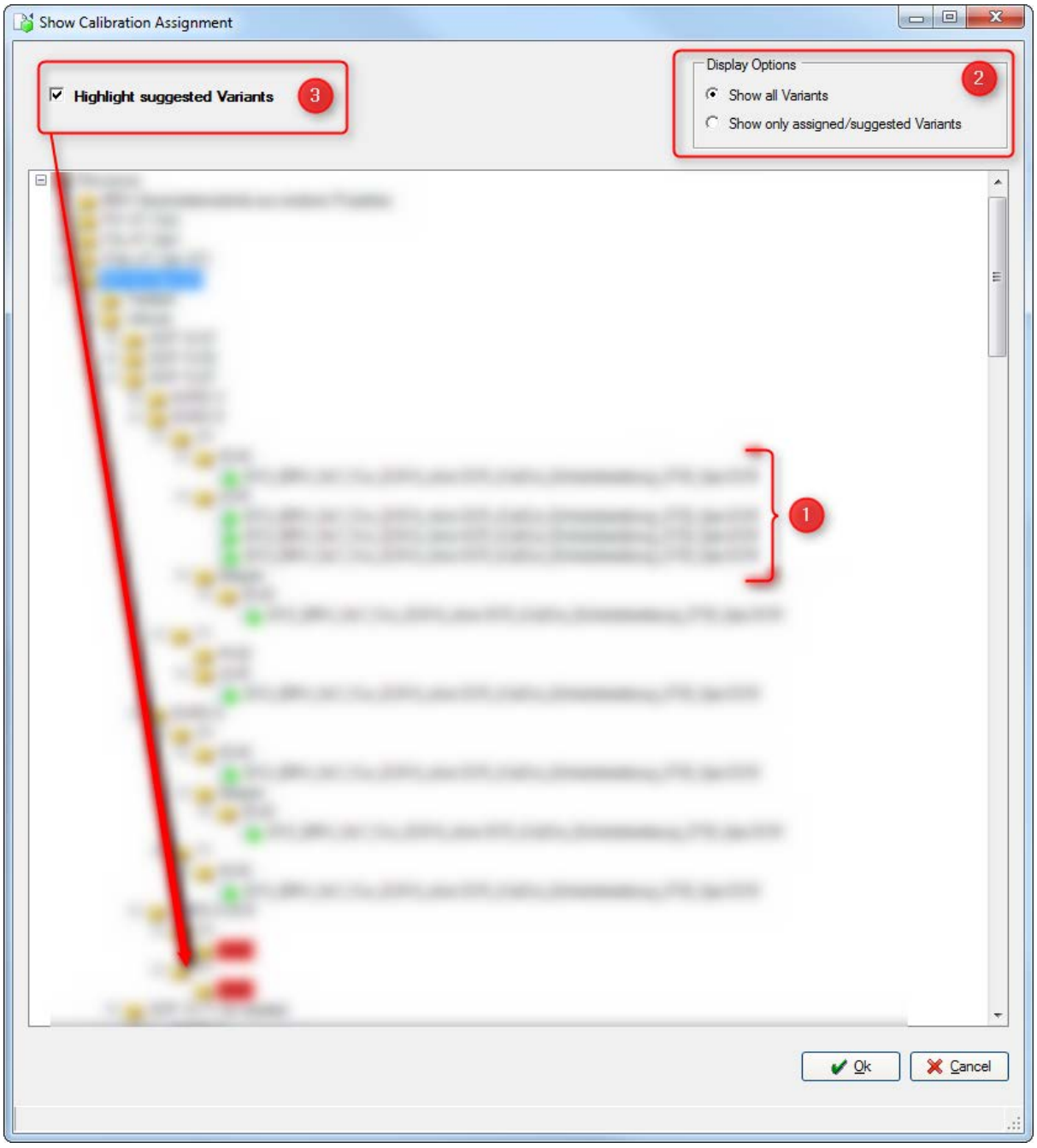


Abbildung 30: Darstellung einer einzelnen Transaktion.

Im Dialog aus Abbildung 30 wird visualisiert, in welche Varianten eine Calibration abgegeben wurde. Die Funktionen dieses Dialoges sind:

1. Anzeige aller Calibrations einer Transaktion und in welche Varianten diese abgegeben wurden anhand einer Baumansicht
2. Ist eine Filteroption, die es erlaubt, nur die zugeordneten bzw. vorgeschlagenen Varianten anzuzeigen um einer besseren Überblick zu bieten

3. Farbliche Hervorhebung jener Varianten, die nach der in Kapitel 4.2.2 beschriebenen *Associative Closure* in der aktuellen Abgabe fehlen

7 Conclusio und Ausblick

Zusammenfassend kann gesagt werden, dass ein Data Mining Prozessmodell eine gute Hilfestellung für das Abarbeiten einer Problemstellung in diesem Bereich bietet. Die unterschiedlichen Prozessmodelle ähneln sich in der Grundstruktur stark. Die Auswahl des entsprechenden Modells ist zwar eher eine Geschmacksfrage, dennoch empfiehlt sich für diese Form der Problemstellung ein gut strukturiertes Vorgehen in Form eines Prozesses.

Für diese Arbeit wurde das CRISP-DM Prozessmodell gewählt, da es aus einer praktischen Anwendung heraus entstanden ist und die daraus gewonnen Erfahrungen in dieses Modell eingeflossen sind. Eine Herausforderung ist es, aus der Fülle unterschiedlicher Data Mining Methoden, die für die Problemstellung passenden zu bestimmen. Die strukturierte und stufenweise Vorgehensweise CRISP-DM Prozessmodell bietet dafür eine nützliche Hilfestellung.

Das Fundament für das Erstellen guter Modelle stellt einerseits ein entsprechendes Verständnis der Daten und andererseits auch ein Verständnis des betrieblichen Gesichtspunktes hinter den Daten dar. Eine dementsprechend gute und regelmäßige Kommunikation zwischen den unterschiedlichen Interessensgruppen ist auch deshalb von großer Bedeutung, damit die technische Umsetzung nicht in die falsche Richtung geht und das Ergebnis den unternehmerischen Erwartungen letztendlich gerecht wird. Häufig übersteigen die gestellten Anforderungen aber auch die Möglichkeiten der technischen Umsetzung. Durch entsprechende Rückmeldungen können die Anforderungen so angepasst werden, dass sie sowohl technisch umsetzbar als auch unternehmerisch von Nutzen sind.

Aus betrieblicher Sicht gab es für die dargelegte Problemstellung anfänglich die falsche Annahme, dass das Vorhandensein einer großen Datenmenge automatisch ausreichen würde, um jede beliebige Methode des Data Mining anwenden zu können. Erst durch eine entsprechende Kommunikation in Bezug auf die Frage danach, welche Input-Daten die jeweiligen Methoden benötigen und was der Output dieser Methoden ist, konnte gemeinsam ein nutzbringender Anwendungsfall definiert werden. Aus den gewonnen Erfahrungen in der praktischen Umsetzung kann gesagt werden, dass der zeitliche Aufwand für das nötige gegenseitige Verständnis zwischen technischer und betrieblicher Sicht nicht unterschätzt werden sollte.

Aufgrund der Beschaffenheit der Datenquelle wurde die Assoziationsanalyse als Methode zur Lösung der Problemstellung gewählt. Diese erkennt zwar Zusammenhänge in Transaktionen, liefert jedoch in der Praxis eine so umfangreiche Anzahl an Ergebnissen, dass diese nicht direkt verwendbar sind. Die gängigen Lösungsansätze durch Sortieren, Filtern oder Visualisieren der Regeln vermindern dieses Problem zwar, bieten aus Anwendersicht, letztendlich aber keine zufriedenstellende Lösung.

Die Ausreißererkennung in Transaktionsdaten liefert hier einen wesentlich besseren Ansatz, wird jedoch in der Literatur kaum erwähnt. Es gibt viele wissenschaftliche Studien, die sich mit der Ausreißererkennung von numerischen Daten beschäftigen. Obwohl die Ausreißererkennung in Transaktionsdaten ein interessanter Ansatz für unterschiedliche Problemstellungen darstellt, wird diese Methode in sehr wenigen Studien aufgegriffen. Diese Tatsache wird auch in der Arbeit von (Narita & Kitagawa, 2008) kritisch erwähnt.

Weiters sei angemerkt, dass das Maß zur Bestimmung von Ausreißern in Abhängigkeit von der Problemstellung gewählt werden muss. Für die gegebene Problemstellung hat sich das od(t)-Maß aus Kapitel 4.2.2 als passend herausgestellt, welches zum Auffinden von Transaktionen mit fehlenden Werten gute Ergebnisse liefert. Zum Auffinden von Transaktionen mit zu vielen Werten ist dieses Maß jedoch nicht geeignet. Ein möglicher Ansatz, auch diese Anforderung zu erfüllen, wird hier abschließend in Form eines kurzen Ausblicks diskutiert.

Es wurde eine weitere Methode, das FPOF(t)-Maß, zur Ausreißererkennung im Prototyp implementiert. Das FPOF(t)-Maß, welches in Kapitel 4.2.1 beschrieben ist, hat sich nach der Evaluierung als nicht brauchbar herausgestellt. Dieses Maß bewertet Transaktionen, in welchen weniger häufige Muster vorkommen, als Ausreißer, was für die gegebene Problemstellung ungeeignet ist.

Abschließend wurde der Prototyp mit einem realen Projekt evaluiert und er lieferte auch zufriedenstellende Ergebnisse. Außerdem wurde er mit einem künstlichen Projekt getestet um seine Leistungsfähigkeit in unterschiedlichen Situationen beurteilen zu können. Aus technischer Sicht wäre es wünschenswert gewesen den Prototyp noch mit weiteren realen Projekten zu testen, jedoch war es aus Gründen der Vertraulichkeit nicht möglich auf Kundenprojekte Zugriff zu bekommen. Aus diesem Grund wird bereits eine Kooperation mit einem Kunden angestrebt, um bei der Evaluierung auf mehr Testdaten zugreifen zu können. Von betrieblicher Seite kam die Aussage, dass die Ergebnisse des Prototyps zufriedenstellend

seien, da diese die Umsetzbarkeit beweisen. Dies liefere eine gute Argumentation für die angestrebte Kooperation mit einem Kunden.

Eine Schwäche in der Bestimmung von Ausreißern mit der Methode aus Kapitel 4.2.2 war die Tatsache, dass Abgaben, die in zu viele Varianten erfolgen, überhaupt nicht erkannt wurden. Ein Ansatz, diese Ausreißer auch zu erkennen, könnte über Dissoziationsregeln erfolgen. Im ersten Schritt werden Dissoziationsregeln der Form $X \rightarrow \neg Y$ gebildet. Anschließend erfolgt ähnlich wie in Kapitel 4.2.2 die Bestimmung eines Ausreißermaßes, indem die tatsächliche Transaktion t einer anhand von Dissoziationsregeln verkleinerten Transaktion t^- gegenübergestellt wird. Diese Methode wird, in einer Weiterentwicklung des Prototyps, zukünftig erprobt.

Literaturverzeichnis

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Assiciation Rules. *International Conference on Very Large Data Bases* (S. 487-499). San Jose, CA: IBM Almaden Research Center.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, I. A. (1996). *Fast Discovery of Association Rules.* Abgerufen am 07. 02 2014 von Publications of Hannu Toivonen: http://www.cs.helsinki.fi/u/htoivone/pubs/advances.pdf

ASAM. (n.d.). *Association for Standardisation of Automation and Measuring Systems*. Abgerufen am 08. 02 2013 von https://wiki.asam.net/display/STANDARDS/ASAM+MCD-2+MC

Azevedo, A., & Santos, M. F. (2008). KDD, SEMMA AND CRISP-DM: A parallel overview. *IADIS International Conference on Information Systems*, (S. 182-185). Madrid.

Azevedo, P. J., & Jorge, A. M. (2007). Comparing Rule Measures for Predictive Association Rules. *Proceedings of the 18th European conference on Machine Learning* (S. 510 - 517). Warschau: Springer Verlag.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., et al. (2000). *CRISP-DM 1.0 Step-by-step data mining guide.* Abgerufen am 14. Januar 2014 von ftp://ftp.software.ibm.com/software/analytics/spss/support/Modeler/Documentation/14/UserManual/CRISP-DM.pdf

Cleve, J. (2011). *Data Mining.* Abgerufen am 14. 01 2014 von Hochschule Wismar, Fakultät für Wirtschaftswissenschaften: http://www.wi.hs-wismar.de/~cleve/vorl/dmining/dmcolor.pdf

Ester, M., & Sander, J. (2000). *Knowledge Discovery in Databases - Techniken und Anwendungen.* Heidelberg: Springer-Verlag.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. (A. A. Intelligence, Hrsg.) *AI MAGAZINE*, S. 37-54.

Fuchs-Kittowski, K. (2002). *Wissens-Ko-Produktion - Verarbeitung, Verteilung und Entstehung von Informationen in kreativlernenden Organisationen.* Abgerufen am 07.

01 2014 von UNI Leipzig - Institut für Informatik: http://www.informatik.uni-leipzig.de/~graebe/Texte/Fuchs-02.pdf

Gabriel, R., Gluchowski, P., & Pastwa, A. (2009). *Datawarehouse & Data Mining.* Witten: W3L GmbH.

Hahsler, M. (2011). *A Comparison of Commonly Used Interest Measures for Association Rules*. Abgerufen am 03. 03 2014 von http://michael.hahsler.net/research/association_rules/measures.html

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, (S. 1-12). Dallas, USA.

He, Z., Huang, J. Z., & Deng, S. (2005). FP-Outlier: Frequent Pattern Based Outlier Detection. *Computer Science and Information Systems, Vol. 2, No. 1*, (S. 103-118). ComSIS Consortium.

Hipp, J. (2003). *Wissensentdeckung in Datenbanken - Verfahren zur effzienten Regelgenerierung und deren Integration in den Wissensentdeckungsprozeß.* Dissertation, Eberhard-Karls-Universität Tübingen.

Hippner, H., Küsters, U., Meyer, M., & Wilde, K. (2001). *Handbuch Data Mining im Marketing - Knowledge Discovery in Marketing Databases.* Braunschweig: Vieweg & Sohn Verlagsgesellschaft mbH.

IBM-Corporation. (2012). *Vergleich von Tabellen- und Transaktionsdaten*. Abgerufen am 20. 02 2014 von IBM SPSS Dokumentation: http://pic.dhe.ibm.com/infocenter/spssdm/v7r0m0/index.jsp?topic=%2Fcom.ibm.spss.dm.userguide.doc%2Fdms_build_model_association_tabvstrans.htm&lang%3Dde

AVL CRETA™ Calibration Data Management. (n.d.). Abgerufen am 04. 03 2014 von https://www.avl.com/creta

Narita, K., & Kitagawa, H. (2008). Outlier Detection for Transaction Databases using Association Rules. *The Ninth International Conference on Web-Age Information Management* (S. 373 - 380). Zhangjiajie, China: Springer.

Petersohn, H. (2005). *Data Mining - Verfahren, Prozesse, Anwendungsarchitektur.* München: Oldenbourg Wissenschaftsverlag GmbH.

Pyle, D. (1999). *Data Preparation for Data Mining.* San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc.

Storfer, G. (2012). *CRETA Blog*. Abgerufen am 04. 03 2014 von https://www.avl.com/creta-blog

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

University of Waikato. (kein Datum). Abgerufen am 09. 12 2013 von WEKA - Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka/

Vodenicharov, M. (2007). *Visualisierung von Assoziationsregeln mit R.* Abgerufen am 02. 03 2014 von http://michael.hahsler.net/stud/done/vodenicharov/Visualisierung_Assoziationsregeln_Vodenicharov.pdf

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining - Practical Machine Learning Tools and Techniques* (3 ed.). Burlington, MA: Morgan Kaufmann.

Listingverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis