

Interpretierbare Outlier Detection

Bachelorarbeit von Dennis Kasch

Tag der Einreichung: 26.04.17

Gutachter: Prof. Dr. Johannes Fürnkranz



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Dennis Kasch
Matrikelnummer: 1526376
Studiengang: B. Sc. Informatik

Bachelorarbeit
Thema: Interpretierbare Outlier Detection

Tag der Einreichung: 26.04.17

Betreuer: Prof. Dr. Johannes Fürnkranz

Erklärung zur vorliegenden Arbeit

Hiermit versichere ich, die vorliegende Bachelorthesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Sämtliche aus fremden Quellen indirekt oder direkt übernommenen Gedanken sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und wurde noch nicht veröffentlicht.

Ort, Datum

Name, Vorname

Inhaltsverzeichnis

1. Einleitung.....	1
1.1 Motivation.....	1
1.2 Struktur der Arbeit.....	1
2. Grundlagen.....	2
2.1 Knowledge Discovery und Data Mining.....	2
2.2 Outlier Detection.....	3
2.2.1 Outliertypen.....	4
2.2.2 Lernmethoden.....	6
2.2.3 Darstellung.....	7
2.3 Frequent Itemset Mining.....	7
2.3.1 Problembeschreibung.....	7
2.3.2 Suchraum.....	8
2.3.3 Border.....	9
2.3.4 Apriori.....	9
3. Interpretierbare Outlier Detection Algorithmus.....	13
3.1 Ansatz.....	13
3.2 Umsetzung.....	13
3.2.1 Suche nach seltenen Attributsets.....	14
3.2.2 Suchraum.....	15
3.2.3 Outlierbestimmung.....	16
3.2.4 Darstellung der Outlier.....	18
4. Experimente.....	19
4.1 Genauigkeit der Outlierbestimmung.....	19
4.2 Ergebnisse.....	20
4.2.1 Quantitative Ergebnisse.....	21
4.2.2 Qualitative Ergebnisse.....	25
4.3 Schlussfolgerung.....	31
5. Verwandte Arbeiten.....	32
5.1 Finding Intensional Knowledge of Distance-Based Outliers.....	32
5.2 Attribute-wise Learning for Scoring Outliers.....	33
5.3 Frequent Pattern Discovery Method for Outlier Detection.....	35
5.4 Discovering the Minimal Set of Unexpected Patterns.....	35
5.5 Local Outlier Detection with Interpretation.....	36
6. Zusammenfassung und Ausblick.....	37
7. Literaturverzeichnis.....	38

1. Einleitung

In der heutigen Zeit gewinnt die maschinelle Verarbeitung von Daten immer mehr an Bedeutung. Besonders die Generierung von Wissen aus Daten ist dabei eine wichtige Aufgabe. Mit unterschiedlichsten Methoden aus den Bereichen des Data Minings und Machine Learnings werden Algorithmen entwickelt, um dieses Problem lösen. Ein Teilbereich des Data Minings ist das Gebiet der Outlier Detection, das sich mit der Identifikation von ungewöhnlichen Daten oder Datenmustern befasst. Diese Outlier (dt. „Ausreißer“) sind in vielen Bereichen von großem Interesse, weil sie Besonderheiten und kritische Zustände aufzeigen können. Oft wird bei der Outlier Detection jedoch wenig Wert auf die Interpretierbarkeit der Ergebnisse gelegt, wodurch der Informationsgewinn für einen Anwender begrenzt ist. In dieser Arbeit liegt der Fokus daher auf der Entwicklung eines Outlier Detection Algorithmus mit gut interpretierbaren Ergebnissen.

1.1 Motivation

Bei der Wissensvermittlung spielt die Interpretierbarkeit von maschinell erzeugten Ergebnissen eine wichtige Rolle. Je verständlicher Outlier und ihre ungewöhnlichen Strukturen dargestellt werden, desto eher ist es einem menschlichen Anwender möglich die getroffene Auswahl nachzuvollziehen und weitere Kenntnisse über die vorliegenden Daten zu gewinnen. Während in der Vergangenheit viele Algorithmen zur effizienten Outlier Detection entwickelt wurden, legen davon jedoch nur sehr wenige Wert auf gute Interpretierbarkeit der Ergebnisse. Oft erhält ein Anwender nur ein Ranking von Outliern, welches wenig oder gar keinen Aufschluss über die zugrundeliegenden Strukturen von Ausreißern gibt. In dieser Arbeit wird daher ein Ansatz verfolgt, der die Interpretierbarkeit von detektierten Outliern ermöglichen soll. Dafür wird unter der Verwendung von Methoden aus dem Frequent Pattern Mining ein Modell erstellt, das gut interpretierbare Eigenschaften von Outliern beschreibt. Instanzen eines Datensatzes werden dann daraufhin überprüft, inwieweit sie dem Modell entsprechen und erhalten eine passende Bewertung. Durch anschließende Präsentation der am höchsten bewerteten Instanzen, zusammen mit ihren seltensten Eigenschaften, ist es schließlich möglich die Besonderheiten von Outliern nachzuvollziehen. Ein Anwender kann so möglich Unterschiede, Gemeinsamkeiten und Besonderheiten in den detektierten Outliern erkennen und damit erweitertes Wissen über die Daten gewinnen.

1.2 Struktur der Arbeit

In Kapitel 2 werden zunächst wichtige Grundlagen für das Verständnis der Arbeit erläutert. Neben allgemeinen Einführungen in die Themen Knowledge Discovery, Data Mining und Outlier Detection, wird im Detail auf Frequent Itemset Mining eingegangen, welches die Grundlagen für den in Kapitel 3 präsentierten Algorithmus bildet. In Kapitel 4 findet schließlich eine experimentelle Untersuchung des entwickelten Algorithmus statt, bei der sowohl quantitative als auch qualitative Eigenschaften betrachtet werden. In Kapitel 5 werden verwandte Arbeiten präsentiert, bevor schließlich in Kapitel 6 Zusammenfassung und Ausblick formuliert werden.

2. Grundlagen

In diesem Kapitel werden kurze Einführungen in die Themen präsentiert, die für das Verständnis dieser Arbeit wichtig sind. Dazu werden zunächst Knowledge Discovery und Data Mining erläutert, bevor genauer auf Outlier Detection eingegangen wird. Dort werden Zielsetzung und Anwendungsgebiete der Outlier Detection besprochen und es findet eine Unterscheidung zwischen unterschiedlichen Outliertypen und Lernmethoden statt. Anschließend wird das Frequent Itemset Mining im Apriori-Algorithmus erläutert, welches die Grundlage für den in Kapitel 3 entwickelten Algorithmus bildet.

2.1 Knowledge Discovery und Data Mining

Durch die zunehmende Digitalisierung von alltäglichen Vorgängen, werden oft große Mengen an Daten gespeichert, die durch ihre Masse für den Menschen inzwischen unüberschaubar geworden sind. Informationen über Einkäufe im Supermarkt, Geldüberweisungen, Nutzungsverhalten im Internet und unterschiedlichste Vorgänge in Industrie und Handel, werden umfangreich aufgezeichnet, lassen in ihrer Reinform aber wenig Schlüsse auf Zusammenhänge in den Daten zu. Eine Aufbereitung und Analyse derartiger Datenmengen durch menschliche Spezialisten ist oft zeit- und kostenintensiv. Es liegt daher nahe, durch Computersysteme generierte Daten auch mit Hilfe von Computeralgorithmen für die Analyse weiter zu verarbeiten. An dieser Stelle setzen Knowledge Discovery und Data Mining an.

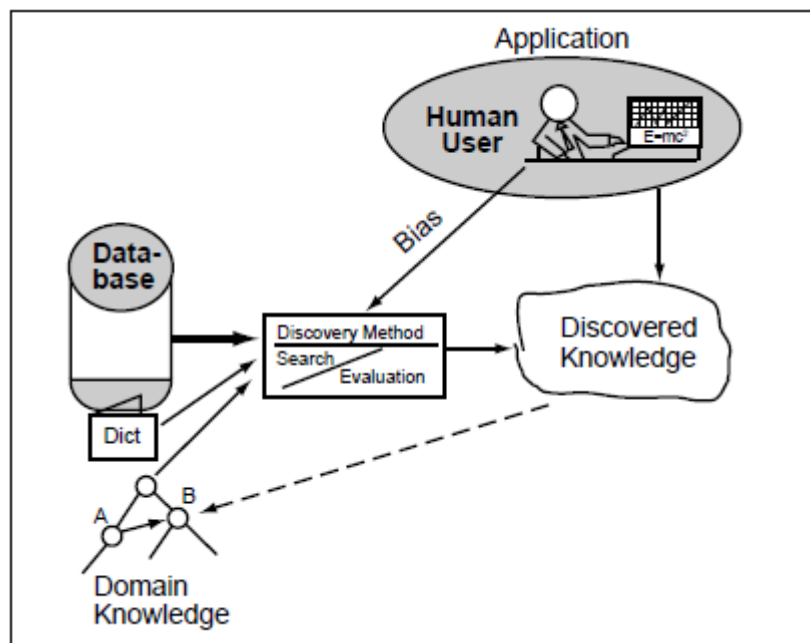


Abbildung 1: Grundstruktur für Knowledge Discovery in Datenbanken

Quelle: [1], Abb. 1

Unter Knowledge Discovery in Datenbanken (KDD) versteht man die Gewinnung von vorher unbekanntem Wissen aus Daten [1]. Es wird dabei nach Mustern in den Daten gesucht, die Aussagen über die Daten oder Teilmengen davon zulassen. Sind die Muster statistisch relevant und drücken interessante Beziehungen in den Daten aus, so können sie als Wissen bezeichnet

werden. Die Ergebnisse eines Programms, welches Daten untersucht und derartige Muster erzeugt, entsprechen demnach entdecktem Wissen. Maße für Relevanz und Interesse können je nach Ansatz sehr unterschiedlich ausfallen.

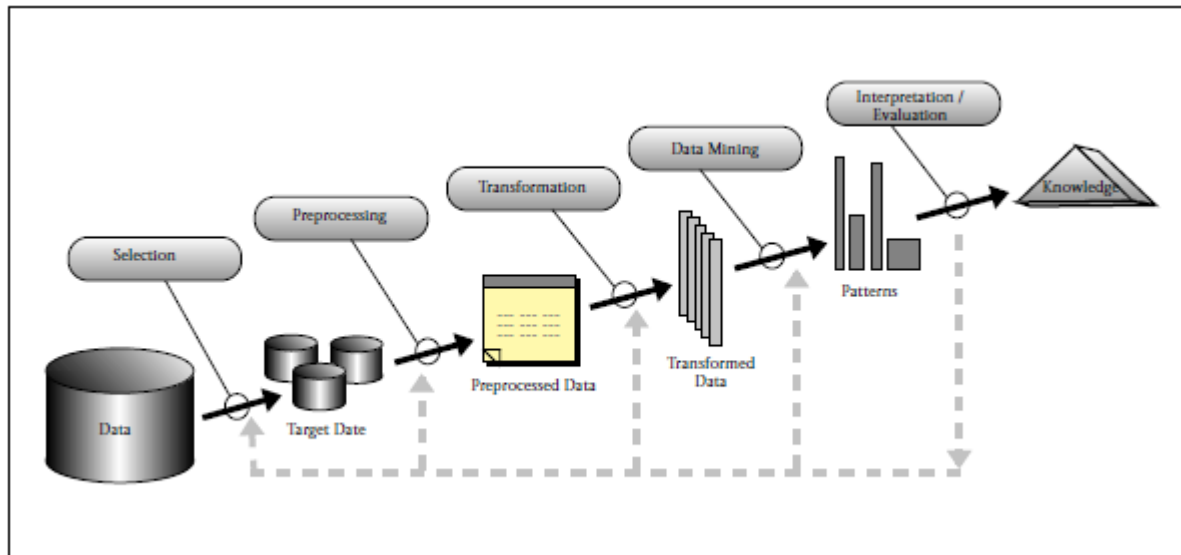


Abbildung 2: Übersicht der einzelnen Schritte im KDD Prozess
Quelle: [2], Abb. 1

Klassischerweise beschreibt KDD den vollständigen Prozess der Wissensgewinnung aus Daten, während Data Mining der reinen Identifikation von Mustern entspricht [2]. Es handelt sich beim Data Mining also um einen bestimmten Schritt, der in der KDD zur Anwendung kommt. Andere Schritte wie Datenselektion, Vorverarbeitung und Interpretation sind unverzichtbar für die Wissensgewinnung, aber nicht Bestandteil der klassischen Definition des Data Minings. Inzwischen werden beide Begriffe jedoch häufig synonym, für den gesamten Prozess der Wissenserkenntnis, verwendet. Wichtig für die Gültigkeit von erzeugten Ergebnissen ist jedoch, dass bei der Anwendung von Data Mining Methoden auch Rücksicht auf Form und Interpretierbarkeit der Ergebnisse genommen wird. Andernfalls können leicht Muster identifiziert werden, die zwar statistisch korrekt sind, aber kaum zusätzliches Wissen generieren.

2.2 Outlier Detection

Outlier Detection (auch „Anomaly Detection“) beschäftigt sich mit der Suche nach ungewöhnlichen Mustern in Daten [6]. Es wird im Allgemeinen davon ausgegangen, dass der Großteil aller Daten in einem Datensatz gewöhnliche Muster aufweist. Trifft dies auf einige Daten nicht zu oder weisen diese überwiegend seltene Muster auf, so können diese als Ausreißer aufgefasst werden. Je nach Anwendungsgebiet tragen Ausreißer oft unterschiedliche Bezeichnungen, wobei Outlier und Anomalien die am häufigsten verwendeten Begriffe sind. Im weiteren Verlauf dieser Arbeit wird der Begriff Outlier für diese Art von Daten verwendet. Eine einheitliche und allgemeingültige Definition von Outliern ist

jedoch schwierig, weshalb bis heute viele Ansätze und Methoden aus unterschiedlichen Fachgebieten existieren [20]. Je nach Anwendungsgebiet und vorliegenden Daten, kann eine andere Auffassung von Outliern von Vorteil sein.

In vielen Anwendungsszenarien sind Outlier von besonderem Interesse, weil sie kritische Zustände in einem System beschreiben. So können Outlier ein Anzeichen für korrumpierte Computer in einem Netzwerk sein [3], bösartige Tumore in einem MRT-Bild [4] oder den Diebstahl von Kreditkarten [5] aufzeigen. Durch die Anwendung von Outlier Detection Algorithmen ist es möglich derartige Zustände und Abläufe zu erkennen und entsprechende Gegenmaßnahmen zu treffen, um gegebenenfalls weiteren Schaden zu verhindern. Ebenso kann Outlier Detection für die Bereinigung von Messfehlern oder Rauschen in einem Datensatz verwendet werden. In anderen Szenarien dient es der Identifikation von ungewöhnlichen Instanzen oder Mustern, die dem Anwender zusätzliches Wissen über die Daten vermitteln.

2.2.1 Outliertypen

Je nach Anwendungsgebiet und Datensatz können unterschiedliche Outlier von Interesse sein. Man unterscheidet Outlier daher in drei Typen, für deren Identifikation meist unterschiedliche Methode verwendet werden.

2.2.1.1 Punkt-Outlier

Als Punkt-Outlier versteht man Outlier, die sich in Bezug auf den gesamten Datensatz von der Mehrheit aller Daten unterscheiden. Als Beispiel dafür können Punkte in einem zweidimensionalen Datensatz dienen. Punkte die viele andere Punkte in ihrer näheren Umgebung haben, werden als normal betrachtet. In Abb. 3 trifft dies auf die Punkte in den Bereichen N_1 und N_2 zu. Die Punkte o_1 und o_2 dagegen weisen jeweils eine große Distanz zu anderen Punkten auf und werden daher als Outlier identifiziert. Ebenso liegen im Bereich O_3 nur wenige Punkte. Je nach Schwellwert für die Anzahl an benötigten Punkten in der Nachbarschaft, können diese entweder als normal oder als Outlier identifiziert werden.

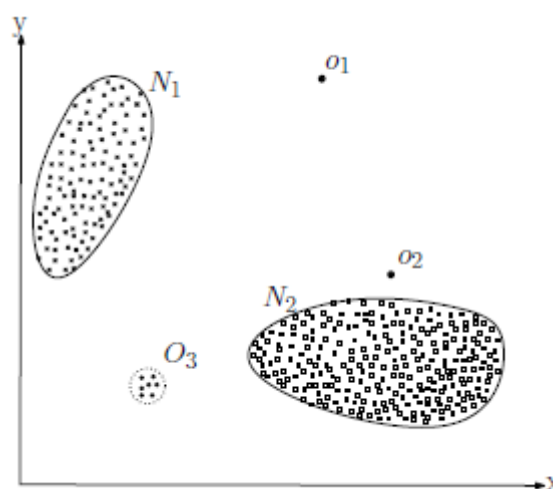


Abbildung 3: Beispiel für Punkt-Outlier
Quelle: [6], Abb. 1

2.2.1.2 Kontext-Outlier

Kontext-Outlier sind Outlier, die nicht über einen gesamten Datensatz als Outlier aufgefasst werden, jedoch in einem speziellem Kontext der Daten. Der Kontext muss dabei in den Daten vorgegeben und klar definiert sein. Man unterscheidet daher zwischen kontextuellen Attributen, die den Kontext einer Instanz beschreiben und Verhaltensattributen, die das Verhalten einer Instanz beschreiben.

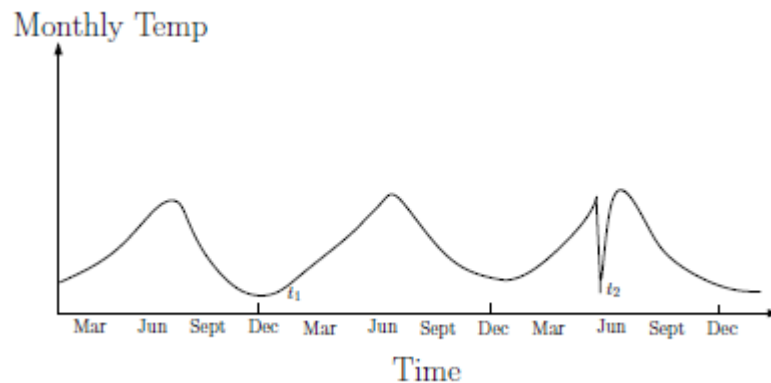


Abbildung 4: Kontext-Outlier in Temperaturmessungen
Quelle: [6], Abb. 3

Ein Beispiel für Kontext-Outlier können Ausreißer in einem Temperaturgraphen sein, der in Abb. 4 zu sehen ist. Über den gesamten Graphen gesehen erfüllt keine der Messungen die Eigenschaften eines Outliers, da an genügend anderen Stellen Temperaturen in einem ähnlichen Bereich auftreten. Betrachtet man jedoch nur die gemessenen Temperaturen im Juni und vergleicht diese miteinander, so zeigt sich für den Punkt t_2 eine ungewöhnlich niedrige Messung. Der Punkt t_2 wird daher als Kontext-Outlier identifiziert. Der Monat einer Messung entspricht dabei einem kontextuellen Attribut, die gemessene Temperatur einem Verhaltensattribut.

2.2.1.3 Kollektive Outlier

Unter kollektive Outliern versteht man eine Ansammlung von Instanzen, deren gemeinsames Auftreten als Outlier aufgefasst werden kann, ohne dass die einzelnen Instanzen als Outlier gelten. Als Beispiel hierfür dienen Aufnahmedaten eines EKGs. Während die in Abbildung 5 rot markierten Messungen einzeln gesehen nicht ungewöhnlich sind, da auch im blauen Bereich ähnliche Werte zu finden sind, ist ihr gehäuftes Auftreten direkt hintereinander jedoch ungewöhnlich und wird als kollektiver Outlier erkannt. Eine solcher Outlier kann in diesem Anwendungsbereich z.B. Indikator für eine Herzrhythmusstörung sein.

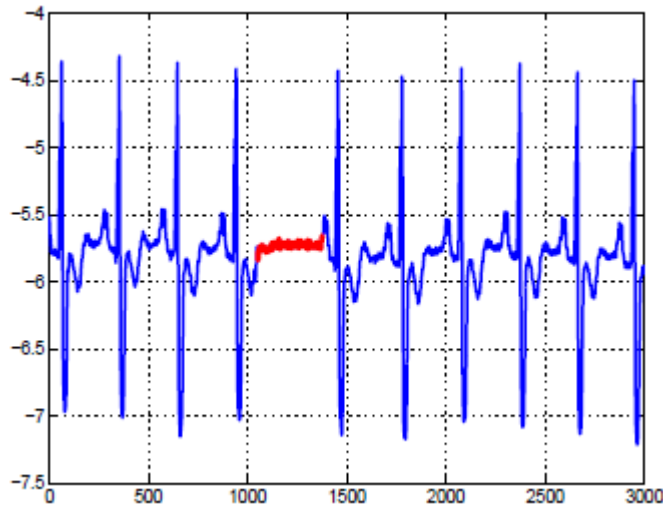


Abbildung 5: kollektive Outlier in einem EKG
Quelle: [6], Abb. 4

2.2.2 Lernmethoden

Oft bedient man sich im Data Mining bei Methoden aus dem Machine Learning. Dort geht es um die maschinelle Generierung von Wissen aus Erfahrung. Dabei werden die vorliegenden Daten als Erfahrung aufgefasst und erzeugte Muster und Gesetzmäßigkeiten gelten als Wissen. Grundstruktur und Zielsetzung sind daher sehr ähnlich zum Data Mining, wodurch große Schnittmengen der beiden Gebiete entstehen.

Wichtig für die Unterscheidung von Methoden des Machine Learnings ist die benötigte Kennzeichnung von Daten, auch als Labeling bezeichnet. Von Menschen durchgeführtes Labeling ist oft aufwändig und setzt meist Expertenwissen im entsprechendem Anwendungsgebiet voraus. Gutes und repräsentatives Labeling ist daher für viele Datensätze nicht vorhanden. Besonders im Gebiet der Outlier Detection ist Labeling oft sehr rar, da Daten die als Outlier gelten meist nur in geringer Zahl auftreten. Je nach Verfügbarkeit von Labels, kommen unterschiedliche Lernmethoden für Outlier Detection zur Anwendung.

2.2.2.1 Überwachte Outlier Detection

Bei überwachten Verfahren zur Outlier Detection wird davon ausgegangen, dass sowohl für normale Daten, als auch für Outlier Labels vorhanden sind. Üblicherweise wird dann ein Modell erstellt, welches Vorhersagen darüber ermöglicht ob neue, zuvor unbekannte Daten zu der Gruppe der Outlier gehört oder nicht. Aufgrund des oft hohen Aufwands für Labeling von Outliern, sind diese Verfahren jedoch in vielen Bereichen nicht anwendbar.

2.2.2.2 Teilüberwachte Outlier Detection

Verfahren dieser Kategorie gehen davon aus, dass das Labeling für normale Instanzen vorliegt. Die übliche Vorgehensweise ist dabei, dass ein Modell für normale Daten erzeugt und auf neue Daten angewendet wird. Entsprechen die neuen Daten nicht den Anforderungen des

Modells, so werden sie als Outlier identifiziert. Durch das Entfallen der Labels von Outliern, sind diese Verfahren für mehr Szenarien anwendbar, als überwachte Outlier Detection.

2.2.2.3 Unüberwachte Outlier Detection

Bei der unüberwachten Outlier Detection ist keinerlei Labeling für die vorliegenden Daten vorhanden. Es wird aber davon ausgegangen, dass Outlier deutlich seltener auftreten als normale Daten. Ist dies nicht der Fall, so sind unüberwachte Verfahren für die vorliegenden Daten oft ungeeignet. Aufgrund des vollständigen Verzichts auf Labels, ist dieser Verfahrenstyp für die meisten Bereiche anwendbar.

2.2.3 Darstellung

Die Darstellung von identifizierten Outliern erfolgt üblicherweise in der Form von Labeling oder Scoring. Beim Labeling erhält jede Instanz entweder eine Kennzeichnung als Outlier oder als normale Instanz. Es entsteht dabei keine weitere Unterscheidung zwischen den einzelnen Outliern. Durch Variation der Parameter können die gefundenen Outlier jedoch variiert werden. Beim Scoring wird dagegen jeder Instanz ein Wert zugewiesen, der angibt zu welchem Grad sie als Outlier gilt. Es entsteht dadurch ein Ranking von Instanzen, das in der Analyse durch einen Schwellwert gekürzt werden kann. Alle Instanzen über dem Wert gelten dann als Outlier. Alle Instanzen unter dem Wert als normal. Ebenso ist es möglich die, bezüglich des Scorings, n besten Outlier zu bestimmen und dem Anwender in entsprechender Reihenfolge aufzulisten.

2.3 Frequent Itemset Mining

Frequent Itemset Mining beschäftigt sich mit der Suche nach Mengen von Objekten in Datensätzen, die häufig auftreten [8]. In vielen Data Mining Anwendungen bilden diese die Grundlage für die Identifikation von interessanten Mustern. So können daraus z.B. Assoziationsregeln, Korrelationen, Sequenzen oder Episoden konstruiert werden. Am häufigsten kommen sie in der Assoziationsanalyse zur Anwendung, bei der nach Regeln gesucht wird, die eine Wenn-Dann-Beziehung in den Daten ausdrücken. So können z.B. in der Warenkorbanalyse Abhängigkeiten zwischen gekauften Artikeln dargestellt werden. Für diese Problemstellung ist der von Agrawal u.a. entwickelte Apriori-Algorithmus [7] ein gängiges Verfahren. Der erste Schritt dieses Algorithmus beschreibt eine effiziente Methode zur Identifikation von häufigen Itemsets in einem Datensatz.

2.3.1 Problembeschreibung

Eine Teilmenge aller Gegenstände, die z.B. in einem Supermarkt käuflich erworben werden können, werden als Itemset bezeichnet. Ein Itemset das k Elemente enthält, heißt k -Itemset. Als Transaktion bezeichnet man dagegen ein Wertepaar aus Transaktionsnummer (TID) und Itemset, welches einen Kauf von Gegenständen eindeutig identifiziert. Eine Transaktionsdatenbank (TDB) besteht aus einer Ansammlung von Transaktionen.

Ein wichtiges Maß im Frequent Itemset Mining ist der Support. Er gibt die Anzahl aller Transaktionen in der Datenbank an, die das entsprechende Itemset enthalten. Liegt der

Support eines Itemsets auf oder über einem definierten Schwellwert, so nennt man es häufiges Itemset. Liegt er darunter, so nennt man es nicht-häufig. Der Schwellwert wird als minimaler Support bezeichnet. Der Support eines Itemsets kann entweder absolut oder relativ zur Anzahl aller Transaktionen angegeben werden.

TID	Transaktion
100	{beer, chips, wine}
200	{beer, chips}
300	{pizza, wine}
400	{chips, pizza}

Tabelle 1: Beispiel für eine TDB
Quelle: [8], Tabelle 1

Betrachtet man die Itemsets in Tabelle 2, so liegen die Itemsets {} und {chips} für einen absoluten, minimalen Support von 2 über dem Schwellwert und sind somit häufig. Die Itemsets {beer, wine} und {beer, chips, wine} liegen dagegen unter dem minimalen Support und sind daher nicht-häufig. Es gilt beim Frequent Itemset Mining alle Itemsets in einer TDB zu finden, die häufig sind.

Itemset	k	abs. Support	rel. Support
{}	0	4	1
{chips}	1	3	0.75
{beer, wine}	2	1	0.25
{beer, chips, wine}	3	1	0.25

Tabelle 2: Itemsets und ihre Supportwerte in der TDB aus Tabelle 1

2.3.2 Suchraum

Der Suchraum einer TDB, mit der Menge aller Items I , umfasst $2^{|I|}$ unterschiedliche Itemsets [8]. Für eine geringe Anzahl von Items ist es leicht möglich alle Itemsets und deren Support zu bestimmen. In der Praxis enthalten TDBs jedoch große Mengen von Items, was schnell dazu führt, dass der vollständige Suchraum und zugehörige Supportwerte nicht mehr in angemessener Zeit durchsucht bzw. bestimmt werden können. Es gilt daher Verfahren zu verwenden, die den Suchraum möglichst reduzieren.

Zur Visualisierung des Suchraums kann die Übermengenrelation verwendet werden. Dabei beginnt man mit dem leeren Itemset und verbindet es mit allen 1-Itemsets. Diese werden wiederum mit all jenen 2-Itemsets verbunden, die eine Übermenge des jeweiligen 1-Itemsets

darstellen. Es werden solange in dieser Weise alle k-Itemsets mit k+1-Itemsets verbunden, bis man schließlich das Itemset erhält, welches alle Items beinhaltet. Dadurch entsteht eine Netzstruktur, die alle Itemsets im Suchraum abbildet.

2.3.3 Border

In der Arbeit von Mannila Heikki und Hannu Toivonen [9] wurde der Begriff der Border eingeführt, der maximal häufige und minimal nicht-häufige Itemsets beschreibt. Damit ein Itemset als Teil der Border gilt, muss es zwei Bedingungen erfüllen:

1. alle Untermengen des Itemsets sind häufig
2. alle Übermengen des Itemsets sind nicht-häufig

Diese Itemsets sind besonders interessant, weil sie den Übergang von häufigen zu nicht-häufigen Itemsets beschreiben. Die Menge aller häufigen Itemsets in der Border wird als positive Border bezeichnet. Die der nicht-häufigen als negative Border.

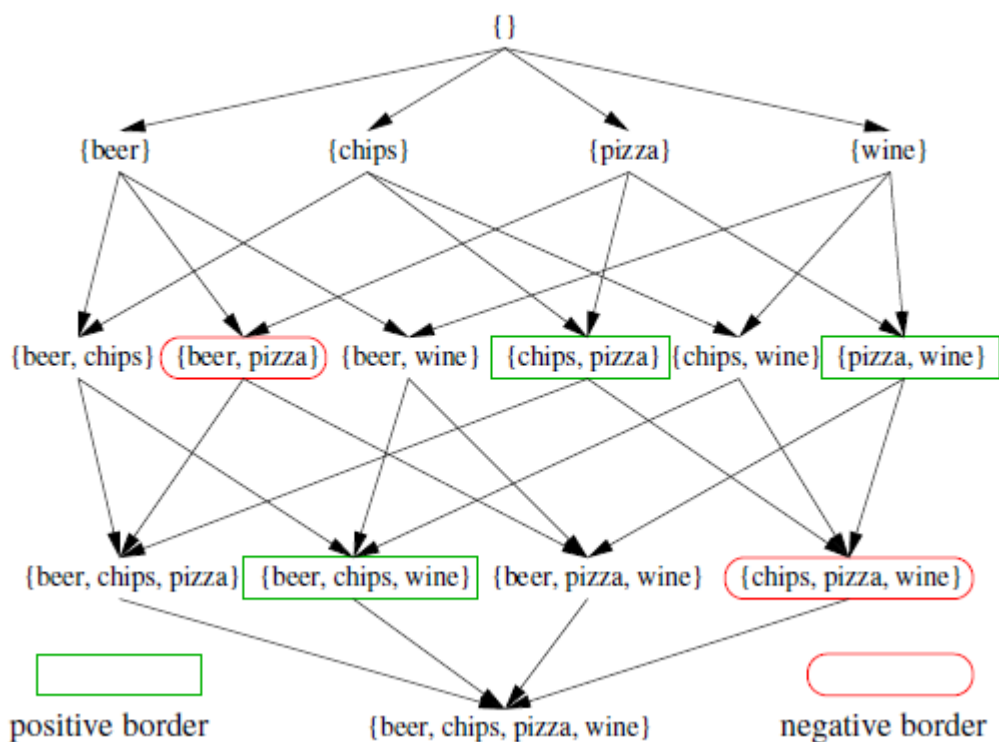


Abbildung 6: positive und negative Border im Suchraum der TDB in Tabelle 1 für min. Support=1, Quelle: [8], Abb. 1

2.3.4 Apriori

Der Apriori-Algorithmus [7] wurde entwickelt, um Assoziationsregeln für die Warenkorbanalyse zu erstellen. Assoziationsregeln sind Regeln der Form $A \Rightarrow B$, wobei A und B nicht-leere Mengen von Objekten sind. Dadurch lassen sich Abhängigkeiten wie

„Kunden die Bier kaufen, kaufen auch Chips“ ausdrücken. Die entsprechende Regel dazu lautet $\{beer\} \Rightarrow \{chips\}$. Grundlage für diese Regeln sind häufige Itemsets, die im ersten Schritt des Algorithmus identifiziert werden. Im zweiten Schritt werden daraus dann entsprechende Regeln konstruiert und anhand eines geeigneten Maßes ausgewählt. Da die Konstruktion von Assoziationsregeln allerdings nicht Gegenstand dieser Arbeit ist, wird hier auf den zweiten Schritt nicht näher eingegangen.

Für die Identifikation von häufigen Itemsets, werden die Itemsets nach und nach erweitert. Dazu wird mit allen 1-Itemsets begonnen und anhand der TDB und dem minimalen Support bestimmt, welche davon häufig sind. Häufige Itemsets werden anschließend zu allen Itemsets mit einem zusätzlichen Element vereinigt. Man bezeichnet die vereinigten Itemsets auch als Kandidaten, da sie potentiell selbst häufig sind. Im nächsten Durchlauf wird dann bestimmt, welche Kandidaten tatsächlich häufig sind und selbst nur häufige Itemsets als Untermenge haben. Diese werden wieder zu neuen Kandidaten mit einem zusätzlichen Element vereinigt. Dieser Vorgang wird solange wiederholt, bis keine neuen Kandidaten für die nächste Iteration mehr erzeugt werden können.

Algorithmus 1 Apriori – Frequent Itemset Mining

```

1: procedure mineFrequentItemsets(instances, minSupport)
2:   k = 1;
3:   S = ∅;
4:   C1 = all single items of instances;
5:   while |Ck| > 0
6:     Sk = all sets of Ck with support ≥ minSupport ;
7:     S = S ∪ Sk ;
8:     Ck+1 = all unions of two sets in Sk with length k+1;
9:     Ck+1 = all sets of Ck+1 that have all subsets of size k in Sk ;
10:    k++;
11:  end while
12: return S;

```

Diese Methode wird ermöglicht durch eine besondere Eigenschaft des Supports. Die Anti-Monotonie des Supports besagt, dass für zwei Itemsets X und Y in einer TDB gilt:

$$X \subseteq Y \Rightarrow \text{support}(Y) \leq \text{support}(X) \quad (1)$$

Das bedeutet dass das Hinzufügen von Items zu einem Itemset niemals zu einem höheren Support für das neue Itemset führen kann. Ebenso können die Übermengen eines nicht-häufigen Itemsets allesamt nur nicht-häufig sein. Umgekehrt müssen die Untermengen eines häufigen Itemsets alle häufig sein. Indem man also ausschließlich häufige Itemsets zu Kandidaten vereinigt, reduziert man die Anzahl der Itemsets die durchsucht werden, erhält

aber noch immer alle häufigen Itemsets.

Wendet man das in Algorithmus 1 beschriebene Frequent Itemset Mining des Apriori-Algorithmus auf die TDB in Tabelle 1 an, so ergeben sich im ersten Schritt, für einen minimalen Support von 1, folgende Mengen:

$$C_1 = \{ \{beer\}, \{chips\}, \{pizza\}, \{wine\} \}$$

$$S_1 = \{ \{beer\}, \{chips\}, \{pizza\}, \{wine\} \}$$

Itemsets S_1 werden zu neuen Kandidaten mit Länge $k+1$ vereinigt. Es ergibt sich für $k=2$:

$$C_2 = \{ \{beer, chips\}, \{beer, pizza\}, \{beer, wine\}, \{chips, pizza\}, \{chips, wine\}, \{pizza, wine\} \}$$

$$S_2 = \{ \{beer, chips\}, \{beer, wine\}, \{chips, pizza\}, \{chips, wine\}, \{pizza, wine\} \}$$

Nach Vereinigung der Itemsets in S_2 , erhält man für $k=3$:

$$C_3 = \{ \{beer, chips, wine\}, \{chips, pizza, wine\} \}$$

$$S_3 = \{ \{beer, chips, wine\} \}$$

Nach der dritten Iteration ist es nicht mehr möglich weitere Kandidaten aus S_3 zu erzeugen. Als Menge aller häufigen Itemsets erhält man somit:

$$S = \{ \{beer\}, \{chips\}, \{pizza\}, \{wine\}, \{beer, chips\}, \{beer, wine\}, \{chips, pizza\}, \{chips, wine\}, \{pizza, wine\}, \{beer, chips, wine\} \}$$

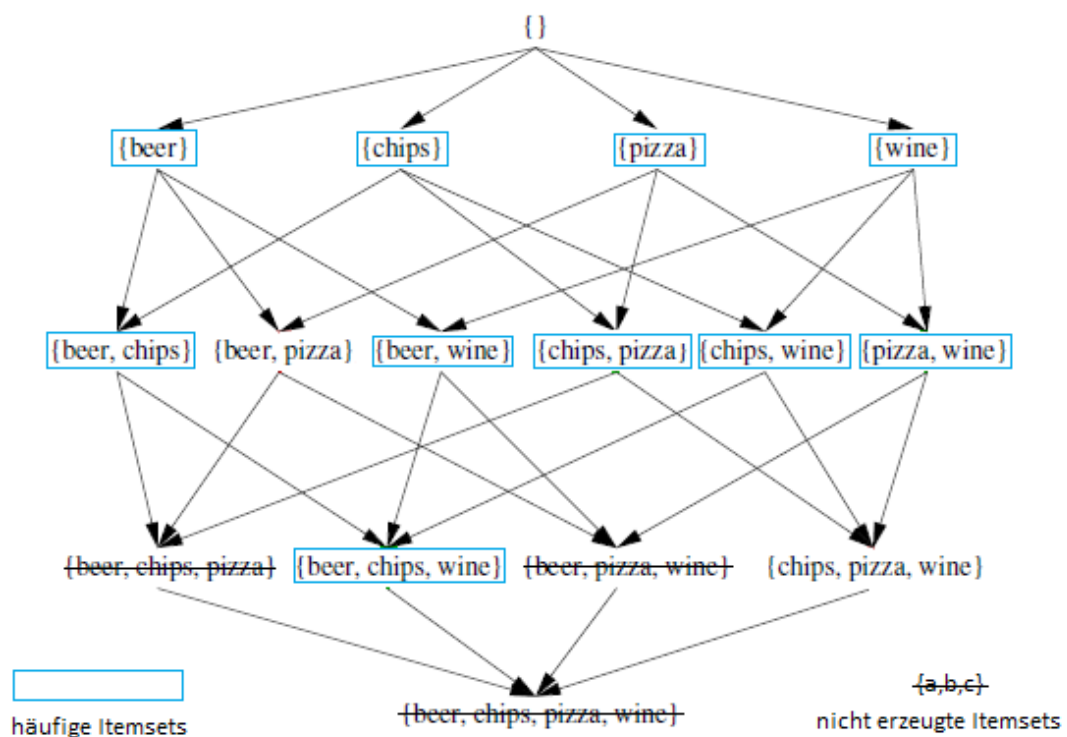


Abbildung 7: erzeugte Itemsets im Apriori-Algorithmus




Abbildung 7 veranschaulicht die erzeugten Itemsets im Apriori-Algorithmus. Die Itemsets {beer, chips, pizza}, {beer, pizza, wine} und {beer, chips, pizza, wine} werden nicht erzeugt, da sie nicht-häufige Untermengen enthalten. Aufgrund der Anti-Monotonie des Supports kann daher von vornherein ausgeschlossen werden, dass sie häufig sind. Die Anzahl an generierten Itemsets kann so für große Datensätze deutlich reduziert werden.

3. Interpretierbare Outlier Detection Algorithmus

In diesem Kapitel wird der im Rahmen der Arbeit entwickelte Algorithmus zur Identifikation von interpretierbaren Outliern vorgestellt. Dazu wird zunächst der grundsätzliche Ansatz für den Algorithmus erläutert, bevor genauer auf die Umsetzung eingegangen wird. Im Detail wird auf die Suche nach seltenen Attributsets eingegangen und das Vorgehen anhand eines Beispiels erläutert. Ebenso wird der Suchraum des Algorithmus betrachtet, so wie auf die Bewertung und Darstellung von Outliern eingegangen.

3.1 Ansatz

Grundlegende Idee des entwickelten interpretierbaren Outlier Detection Algorithmus (IODA) ist, dass eine Auflistung von seltenen Eigenschaften eines Outliers, für die Interpretierbarkeit und damit dem Informationsgewinn eines Anwenders, zuträglich ist. Um derartige Eigenschaften ausfindig zu machen, werden beim IODA Kombinationen von Attributwerten in einem Datensatz als Attributsets aufgefasst. Durch diese Interpretationsweise ist es möglich, die im Apriori-Algorithmus verwendete Methode zur Erzeugung von Itemsets auf die Attribute anzuwenden. Anstatt jedoch häufige Attributsets zu generieren, die als normale Eigenschaften betrachtet werden können, werden nicht-häufige Attributsets erzeugt. Bei einem entsprechenden Support für die Erzeugung der Sets, können diese als seltene Eigenschaften im Datensatz aufgefasst werden und bilden somit die Grundlage für die Identifikation von Outliern im IODA.

Sind die seltenen Attributsets in einem Datensatz bestimmt, lassen sich Instanzen daraufhin untersuchen inwieweit sie seltene Attributsets abdecken. Je mehr seltene Sets eine Instanz enthält, desto mehr seltene Eigenschaften besitzt sie und desto eher entspricht sie einem Outlier. Weiterhin sollten die Attributsets eines Outliers einen möglichst geringen Durchschnittssupport aufweisen. Auf Grundlage dieser beiden Maße wird schließlich ein Ranking konstruiert, das angibt welche Instanzen am ehesten einem Outlier entsprechen. Die zugehörigen Attributsets eines Outliers dienen dabei als Veranschaulichung für seine Besonderheiten. Durch Vergleich von Instanzen und ihren unterschiedlichen Attributsets, sollte es einem Anwender möglich sein allgemeineres Wissen über die Eigenschaften von Outliern abzuleiten.

3.2 Umsetzung

Der IODA wurde als Ranker in der Weka Java API [10] implementiert. Dabei wurde die vorhandene Implementierung des Apriori-Algorithmus verwendet und entsprechend verändert, um seltene Attributsets zu identifizieren. Der IODA folgt dabei einem unüberwachten Lernansatz, der ohne Labels für Outlier auskommt. Es werden Punkt-Outlier erkannt, die sich über den gesamten Datensatz betrachtet hervorheben. Mit geringen Anpassungen sollte sich der Algorithmus aber auch für die Identifizierung von lokalen Outliern verwenden lassen.

3.2.1 Suche nach seltenen Attributsets

Den Kern des IODA bildet die, auf dem Frequent Pattern Mining des Apriori-Algorithmus basierende, Suche nach Attributsets. Während beim Apriori-Algorithmus häufige Itemsets entstehen, werden beim IODA jedoch seltene Attributsets erzeugt. Dies geschieht, indem weiterhin häufige Sets zu neuen Kandidatensets vereinigt werden. Anstatt jedoch diese zu sammeln, werden nicht-häufige Sets akkumuliert. Die Vereinigung von seltenen Attributsets ist für die Suche nicht sinnvoll, da aufgrund der Anti-Monotonie des Supports nur gleich seltene oder seltenere Sets entstehen können. Diese beschreiben seltene Eigenschaften, die bereits durch die vorherigen, seltenen Sets besser beschrieben werden. Bei der Vereinigung von häufigen Attributsets entstehen dagegen weiterhin Sets, die nicht-häufig und somit für die Identifikation von Outliern interessant sind. Anstatt alle nicht-häufigen Sets zu erzeugen, werden also nur jene erzeugt, deren Untermengen häufig sind.

In Zeile 3 der in Algorithmus 2 beschriebenen Attributsuche, werden die Kandidaten zunächst mit allen einzelnen Attributen und all ihren möglichen Werten im Datensatz initialisiert. Anschließend werden diese in einer Schleife in häufige und nicht-häufige Sets unterteilt. S_k enthält die Menge aller häufigen, R_k die Menge aller nicht-häufigen Sets der Länge k . Die häufigen Sets werden in Zeile 8 zu neuen Kandidaten für die nächste Iteration kombiniert. Es werden dabei nur häufige Sets der Länge k zu Sets der Länge $k+1$ vereinigt. Von diesen werden in Zeile 9 jene Sets entfernt, deren Untermengen mit Länge k nicht alle in S_k enthalten sind. Diese Sets können aufgrund der Anti-Monotonie des Supports selbst nicht häufig sein. Diese Schritte werden solange wiederholt, bis keine neuen Attributsets mehr aus S_k kombiniert werden können. Die Schleife wird beendet und die Menge aller nicht-häufigen bzw. seltenen Attributsets wird zurückgegeben.

Algorithmus 2 - Suche nach seltenen Attributsets

procedure findRareAttributeSets(instances, minSupport)

```
1:  k = 1;
2:  R = ∅;
3:  C1 = all single attributes with all different values in instances;
4:  while |Ck| > 0
5:    Sk = all sets of Ck with support < minSupport ;
6:    Rk = all sets of Ck with support ≥ minSupport ;
7:    R = R ∪ Rk ;
8:    Ck+1 = all unions of two sets in Sk with length k+1;
9:    Ck+1 = Ck+1 \ sets that do not have all subsets of size k in Sk ;
10:   k++;
11: endwhile
12: return R;
```

Betrachtet man den Beispieldatensatz in Tabelle 3, so werden zunächst alle Attributwerte zur Initialisierung verwendet. Dies ist wichtig, da z.B. auch das Fehlen von Eigenschaften für die Identifikation von Outliern relevant sein kann. Für den initialen Schritt und die erste Iteration ergeben sich bei einem minimalen Support von 2 folgende Mengen:

$$C_1 = \{ \{M\}, \{\neg M\}, \{E\}, \{\neg E\}, \{L\} \}$$

$$S_1 = \{ \{M\}, \{\neg M\}, \{E\}, \{\neg E\}, \{L\} \}$$

$$R_1 = \emptyset$$

	wolf	scorpion	snake	platypus	frog	cow
mammal (M)	true	false	false	true	false	true
eggs (E)	false	false	true	true	true	false
lungs (L)	true	true	true	true	true	true

Tabelle 3: Beispieldatensatz mit 6 Instanzen und 3 Attributen

In der ersten Iteration werden daher noch keine seltenen Attributsets identifiziert. Aus S_1 werden nun neue Kandidaten für die zweite Iteration generiert. Es ergeben sich folgende Mengen:

$$C_2 = \{ \{M, E\}, \{M, \neg E\}, \{M, L\}, \{\neg M, E\}, \{\neg M, \neg E\}, \{\neg M, L\}, \{E, L\}, \{\neg E, L\} \}$$

$$S_2 = \{ \{M, \neg E\}, \{M, L\}, \{\neg M, E\}, \{\neg M, L\}, \{E, L\}, \{\neg E, L\} \}$$

$$R_2 = \{ \{M, E\}, \{\neg M, \neg E\} \}$$

R_2 wird der Gesamtmenge aller seltenen Attributsets R hinzugefügt. Aus S_2 werden nun wieder Kandidaten generiert. Es ergeben sich folgende Mengen für die dritte Iteration:

$$C_3 = \{ \{M, \neg E, L\}, \{\neg M, E, L\} \}$$

$$S_3 = \{ \{M, \neg E, L\}, \{\neg M, E, L\} \}$$

$$R_3 = \emptyset$$

Da aus S_3 nun keine weiteren Kandidaten mehr erzeugt werden können, wird die Schleife beendet. Für den Beispieldatensatz werden also die seltenen Attributsets $R = R_2 = \{ \{M, E\}, \{\neg M, \neg E\} \}$ gefunden. Die Attributsets $\{M, E, L\}$ und $\{\neg M, \neg E, L\}$ werden nicht erzeugt, da diese Sets nicht ausschließlich häufige Untermengen haben. Es erscheint sinnvoll diese Art von nicht-häufigen Sets nicht zu erzeugen, da die Untermengen $\{M, E\}$ und $\{\neg M, \neg E\}$ bereits die zugrundeliegende, seltene Eigenschaften beschreiben. Jedes damit kombinierte Attribut ist aufgrund der Anti-Monotonie des Supports zwar ebenfalls nicht-häufig, bringt jedoch keinen Informationsgewinn mit sich.

3.2.2 Suchraum

Betrachtet man den Suchraum der seltenen Attributsets, so fällt auf dass die Anzahl aller erzeugbaren Attributsets größer ist, als die von Itemsets im Apriori-Algorithmus. Der Grund dafür ist, dass auch das Fehlen einer Eigenschaft berücksichtigt wird. Bei einer Menge von binären Attributen M , ergeben sich für den Suchraum $3^{|M|}$ mögliche Attributsets. Der Suchraum kann jedoch durch das Verwerfen von Werten, die im Datensatz nicht vorkommen

und die bereits im Apriori-Algorithmus verwendete Anti-Monotonie des Supports beschränkt werden. Anstatt des vollständigen Suchraums mit $3^{|\mathcal{I}|} = 27$ Sets, werden für das Beispiel in Tabelle 1 nur die in Abb. 6 dargestellten 16 Sets erzeugt. Viele Sets können bereits durch das Fehlen des Wertes $\{\neg L\}$ im Datensatz ausgeschlossen werden. Die Sets $\{M, E, L\}$ und $\{\neg M, \neg E, L\}$ werden ebenfalls nicht erzeugt, da beide mindestens eine nicht-häufige Untermenge besitzen.

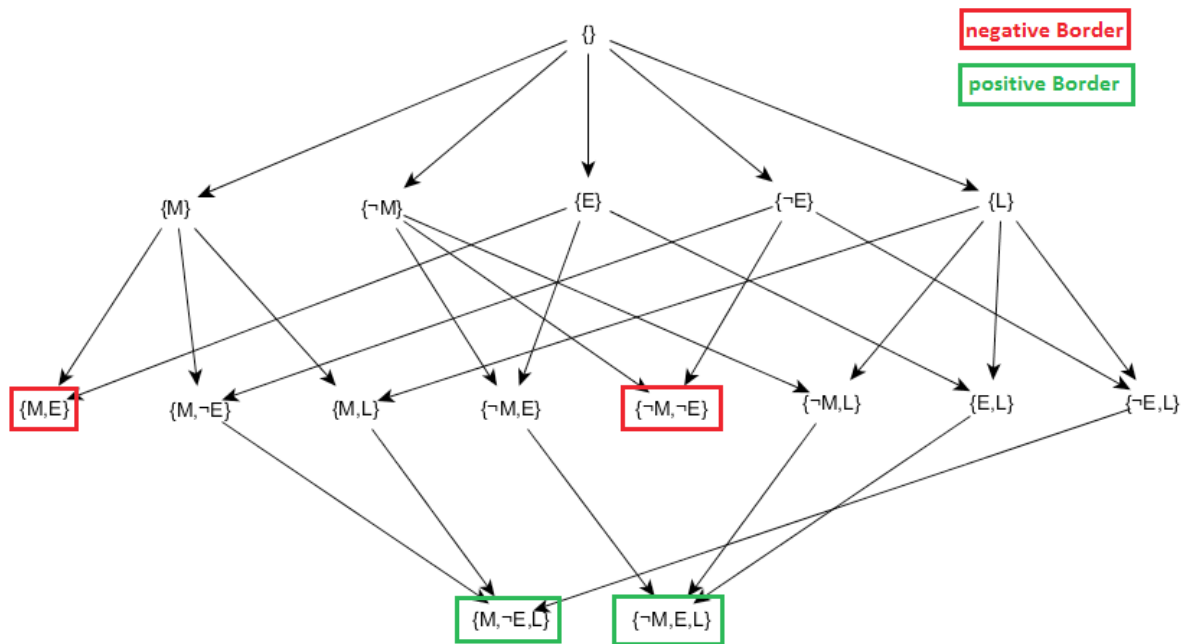


Abbildung 6: Suchraum für seltene Attributsets im IODA

Für die Border ergeben sich dabei die Attributsets $\{M, E\}$, $\{\neg M, \neg E\}$, $\{M, \neg E, L\}$ und $\{\neg M, E, L\}$, wobei $\{M, E\}$ und $\{\neg M, \neg E\}$ die negative Border und $\{M, \neg E, L\}$ und $\{\neg M, E, L\}$ die positive Border stellen. Das heißt die in 3.2.1 identifizierten Attributsets liegen alle in der negativen Border. Grund dafür ist, dass die leere Menge zunächst immer häufig ist und daher alle einelementigen Attributsets ausschließlich häufige Untermengen haben. Weiterhin werden sämtliche Kandidaten aus häufigen Untermengen vereinigt und jegliche Kandidaten entfernt, die eine nicht-häufige Untermenge besitzen. Daraus folgt, dass jedes erzeugte Set ausschließlich häufige Untermengen haben kann, womit die erste Bedingung der Border erfüllt ist. Zusätzlich werden nur nicht-häufige Sets identifiziert. Das bedeutet, dass sie aufgrund der Anti-Monotonie des Supports nur nicht-häufige Übermengen haben können, wodurch die zweite Bedingung der Border erfüllt ist. Die identifizierten Sets müssen daher alle in der Border liegen. Aufgrund ihrer Nicht-Häufigkeit, muss es die negative Border sein.

3.2.3 Outlierbestimmung

Ist die Menge aller relevanten Attributsets A für den Datensatz bestimmt, so lassen sie sich für die Identifikation von Outliern verwenden. Dabei wird jedes gefundene Attributset als seltene

Eigenschaft aufgefasst. Jede Instanz wird dann darauf untersucht, inwieweit sie seltene Eigenschaften besitzt. Je mehr seltene Eigenschaften sie aufweist, desto höher ist ihr Outlierscore, da ihr Verhalten die meisten Abweichungen von der Norm beschreibt.

Um den Score einer Instanz i zu bestimmen, wird die Anzahl aller relevanten Attributsets, die durch i abgedeckt werden, bestimmt und durch die Anzahl alle relevanten Attributsets im gesamten Datensatz geteilt. Man erhält so einen Score, der im Intervall $[0,1]$ liegt. Für die Menge aller möglichen Attributsets A und die Menge aller durch i abgedeckten Attributsets A_i , ergibt sich folgender naiver Score:

$$naiveScore(i) := \frac{|A_i|}{|A|} \quad (2)$$

Es kommt jedoch vor, dass zwei Instanzen die gleiche Anzahl an seltenen Attributsets aufweisen. Um diese noch weiter unterscheiden zu können, wird der Durchschnittssupport der zugehörigen Attributsets in die Berechnung des Scores miteinbezogen. Die zugrundeliegende Idee ist, dass eine Instanz mit gleich vielen, aber durchschnittlich selteneren Auffälligkeiten ein besserer Outlier ist. Da der Durchschnittssupport nur als sekundäres Maß für die Unterscheidung zwischen Instanzen mit gleicher Anzahl an seltenen Attributsets dient, wird er ebenfalls normalisiert und von der Anzahl der Sets abgezogen. Bei der Menge aller Instanzen I , ergibt sich die Formel:

$$NAS(A_i) := \frac{\sum_{r \in A_i} support(r)}{|I| * |A_i|}, \text{ wobei } S=r \in A_i \quad (3)$$

Man erhält damit folgenden verbesserten Score, mit Unterscheidung von Instanzen mit gleicher Anzahl an Sets, aber unterschiedlichem Durchschnittssupport:

$$betterScore(i) := \frac{|A_i| - NAS(A_i)}{|A|} \quad (4)$$

Betrachtet man wieder das Beispiel aus Tabelle 3 und identifiziert die seltenen Attributsets für einen minimalen Support von 3, so ergeben sich die Sets $\{M, E\}$, $\{M, \neg E\}$, $\{\neg M, E\}$ und $\{\neg M, \neg E\}$. Da alle Instanzen eines der seltenen Sets abdecken, ergibt sich nach Formel 3 für alle ein identischer Score von 0,25. Verwendet man jedoch den in Formel 4 beschriebenen Score, werden die Unterschiede im Support der Sets berücksichtigt und man erhält die in Tabelle 4 aufgeführten Werte. Die Instanzen „scorpion“ und „platypus“ sind in diesem Fall, trotz gleicher Anzahl an seltenen Eigenschaften, bessere Outlier als die restlichen Instanzen.

Instanz	wolf	scorpion	snake	platypus	frog	cow
betterScore	0,1667	0,2083	0,1667	0,2083	0,1667	0,1667

Tabelle 4: betterScore-Werte für min. Support=3

3.2.4 Darstellung der Outlier

Von einer Auflistung aller Instanzen mit ihrem jeweiligen Outlierscore ist abzusehen, da in der Regel nur die n Instanzen mit dem höchsten Score von Interesse sind. Diese können wiederum viele seltene Attributsets aufweisen. Es ist daher sinnvoll nur bestimmte Sets für die Darstellung auszuwählen. Hierbei scheinen die Sets mit dem geringsten Support am interessantesten, da diese von den wenigsten Instanzen geteilt werden und somit ein potentielles Alleinstellungsmerkmal für Outlier darstellen. Für die Reduktion der Sets kann entweder ein erneuter Schwellwert für den Support der Sets oder die Anzahl an anzuzeigenden Sets mit geringstem Support festgelegt werden.

4. Experimente

In diesem Kapitel werden die Auswirkungen der Parameterwahl, auf die Ergebnisse des IODA untersucht. Als obere Grenze für die Länge von Attributsets wurde für alle Datensätze einheitlich der Wert 5 festgelegt, da die Erzeugung von längeren Sets in der Regel sehr zeitintensiv ist. Weiterhin sind längere Sets als Erklärung für einen Outlier oft nur schwer interpretierbar. Als veränderliche Größe bleibt damit der initiale Support, der hier in relativer Form angegeben wird, um die Vergleichbarkeit zwischen den Datensätzen zu gewährleisten.

Die für die Experimente verwendeten Datensätze stammen aus dem UCI Machine Learning Repository [11] und werden normalerweise für Klassifikationsverfahren verwendet. Um diese auf das Problem der Outlier Detection anzupassen, wurde nach einem Ansatz von Harkins u.a. [12] jeweils die Klasse mit der geringsten Anzahl an Instanzen als Outlierklasse definiert. Sofern nötig, wurde diese zusätzlich reduziert, um die Anzahl an Outliern im Datensatz auf etwa 5% zu bringen. Außerdem wurden numerische Attribute unter Verwendung der automatischen Diskretisierungsfunktion des Weka-Explorers [13] in nominale Attribute umgewandelt, um ihre Verwendung im IODA zu ermöglichen. Nach Anpassung der Datensätze ergeben sich folgende Werte:

	Breast-Cancer	Iris	Labor	Vehicle	Votes	Zoo
Attribute	9	4	16	18	16	16
Instanzen	211	105	39	681	279	101
Outlier	10	5	2	34	12	4

Tabelle 5: Zusammenfassung der verwendeten Datensätze

4.1 Genauigkeit der Outlierbestimmung

Um zu bestimmen wie präzise das Ranking der Instanzen ist, wurden zwei Maße verwendet: Average Precision (AP) und Area Under ROC Curve (AUC). Um die AP zu bestimmen, benötigt man das Hilfsmaß Precision at k ($P@k$), welches bestimmt wie präzise der Algorithmus in den besten k Ergebnissen ist. Es gilt:

$$P@k = \frac{\text{number of actual outliers in top } k \text{ results}}{k} \quad (5)$$

Die AP lässt sich damit folgendermaßen berechnen:

$$AP = \frac{\sum_{k=1}^n (P@k * \text{outlier}(k))}{\text{number of all outliers}} \quad (6)$$

,mit $\text{outlier}(k) = 1$ wenn Instanz auf Position k ist Outlier, sonst 0

Für die AUC wird zunächst die Receiver Operating Characteristic Kurve (ROC-Kurve) bestimmt. Diese wird verwendet, um Ranker bezüglich verschiedener Kostenmodelle zu untersuchen. Dabei werden die falsch positiven Ergebnisse in Verhältnis zu den echt positiven Ergebnissen gesetzt und in einem normalisierten Graphen dargestellt. Um die AUC zu erhalten, wird schließlich die Fläche unter der ROC-Kurve berechnet. Die AUC lässt dann Aussagen darüber zu, wie gut die Instanzen gerankt werden.

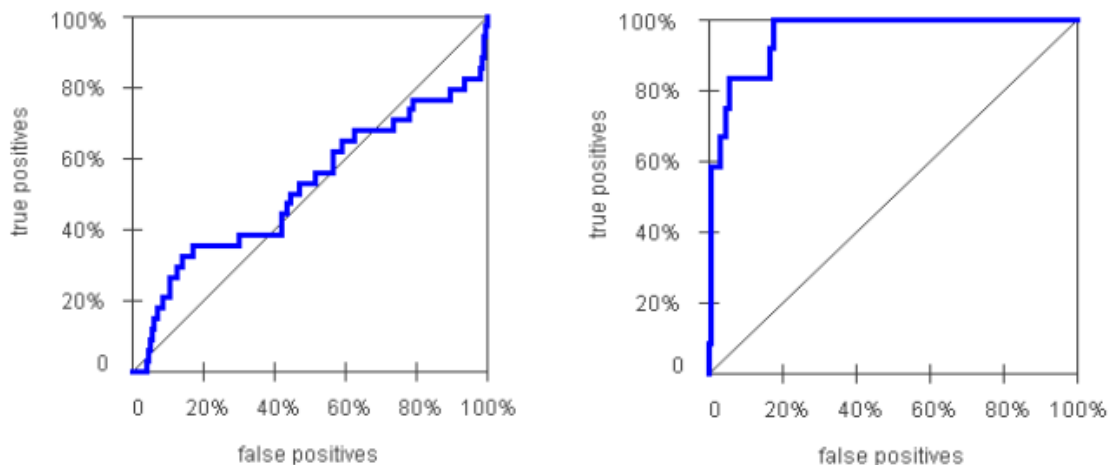


Abbildung 7: links: schlechte ROC-Kurve mit Verlauf an der Diagonalen, rechts: gute ROC-Kurve mit konvexem Verlauf

Um die ROC-Kurve für einen Ranker zu erhalten, werden die echt positiven Ergebnisse auf der X-Achse und die falsch positiven Ergebnisse auf der Y-Achse in Prozent aufgetragen. Dann beginnt man bei derjenigen Instanz mit dem höchsten Score und arbeitet sich das Ranking hinunter. Ist die jeweilige Instanz tatsächlich ein Outlier, so beschreibt der Graph einen Schritt um $100/P$ nach oben, wobei P der Anzahl aller echten Outlier entspricht. Ist sie kein Outlier, so bewegt sich der Graph um $100/N$ nach rechts, mit $N = \text{Anzahl aller falsch positiven Outlier}$. Je weiter oben die tatsächlichen Outlier im Ranking stehen, desto früher steigt die Kurve an. Eine konvexe ROC-Kurve ist daher ein Zeichen für einen guten Ranker. Eine konkave Kurve spricht dagegen für einen Ranker, der die Positionen schlechter vorhersagt, als eine zufällige Auswahl. Eine Kurve nah an der Diagonalen entspricht einem eher zufälligen Ranking der Instanzen.

4.2 Ergebnisse

Die Ergebnisse werden in zwei Teilen präsentiert. Im quantitativen Teil wird gezeigt, wie gut der IODA in Abhängigkeit des initialen Supports Outlier erkennt. Dafür werden die Maße Average Precision und AUC verwendet. Zusätzlich werden unterschiedliche Durchschnittswerte der erzeugten Sets dargestellt, um daraus eventuell Schlüsse auf die Wahl eines guten initialen Supports ziehen zu können. Im qualitativen Teil, der den Schwerpunkt der Experimente darstellt, werden die bezüglich AP und AUC besten Ergebnisse betrachtet und auf ihre Interpretierbarkeit hin untersucht. In der Praxis kann die Anzahl der

interessanten Outlier und ihrer zugehörigen, seltenen Attributsets stark variieren. Um die Ergebnisse überschaubar zu halten, werden die 4 besten Outlier und ihre 5 seltensten Attributsets präsentiert. Dies entspricht einer stichprobenartigen Darstellung, die jedoch je nach Anwendungsszenario beliebig erweitert werden kann. Durch Betrachtung der Sets und Vergleich der dargestellten Eigenschaften der Outlier, wird dann gezeigt inwieweit Interpretationsmöglichkeiten für die gefundenen Outlier gegeben sind.

4.2.1 Quantitative Ergebnisse

Betrachtet man die Durchschnittslänge aller im IODA erzeugten Attributsets in Abb. 8, so ist ein stetiger Abwärtstrend zu beobachten. Dies ergibt sich daraus, dass mit steigendem initialen Support immer mehr kurze Sets als selten betrachtet werden und somit nicht mehr für die Erzeugung längerer Sets zur Verfügung stehen. Spätestens bei einem initialen Support von 100% werden alle einzelnen Attribute als selten betrachtet und es entstehen nur noch Sets der Länge 1. Die durchschnittliche Setlänge konvergiert daher mit zunehmendem Support immer gegen 1.

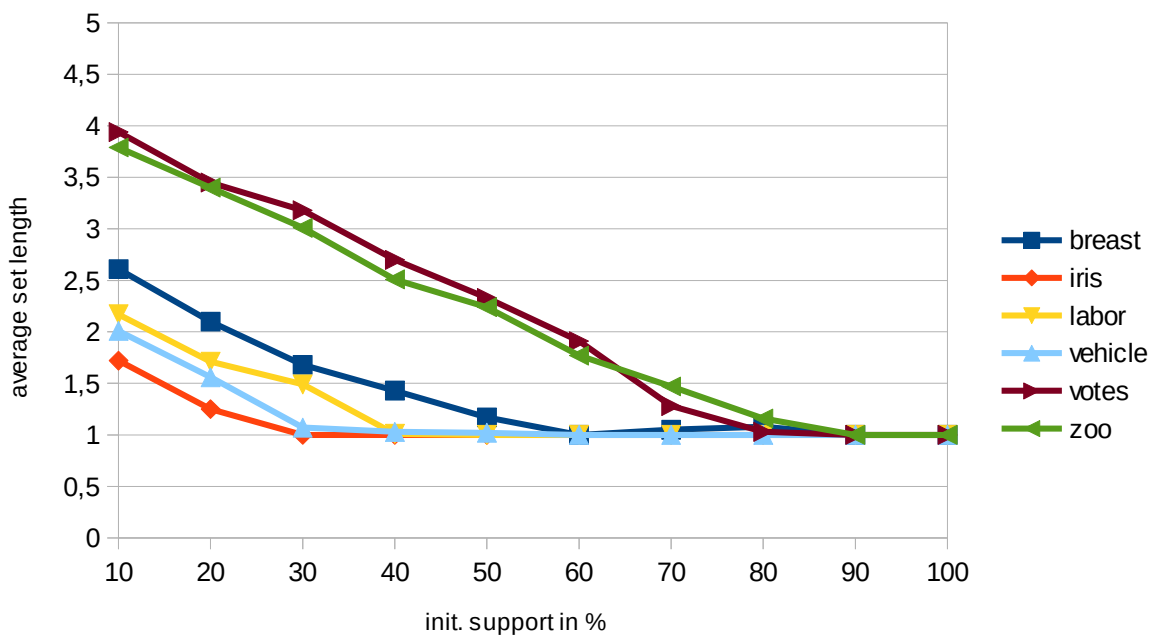


Abbildung 8: Durchschnittslänge der im IODA erzeugten Sets

Der durchschnittliche Support der Sets (Abb. 9) nimmt dagegen mit höherem initialen Support tendenziell zu. Dies scheint einleuchtend, da immer mehr Sets mit höherem Support als selten betrachtet werden, wodurch der Durchschnittssupport steigt. Dennoch gibt es im späteren Verlauf oft noch kleinere Einbrüche. Grund dafür sind Attributsets von größerer Länge, mit höherem Support. Diese werden an dieser Stelle, aufgrund der vielen bereits als selten betrachteten Sets mit kürzerer Länge, nicht mehr erzeugt. Gegen Ende hin wirkt die

Anzahl der einzelnen Attribute mit hohem Support diesem Effekt jedoch wieder entgegen, wodurch ein erneuter Anstieg entsteht.

Im Graphen der AP (Abb. 10) sind unterschiedliche Ergebnisse für die Datensätze zu beobachten. So zeigen Iris- und Labor-Datensatz bei niedrigem init. Support auch niedrige AP-Werte, steigen dann an und landen schließlich bei einem konstanten Wert. Votes- und Breast-Cancer-Datensatz nehmen einen ähnlichen Verlauf. Der Votes-Datensatz steigt bei 80% auf seinen Höhepunkt, bevor er für 90% und 100% konstant bleibt. Breast-Cancer nimmt dagegen seinen höchsten Wert bei 60% an, sinkt dann aber wieder leicht ab und bleibt für 90% und 100% ebenfalls konstant. Für den Vehicle-Datensatz liegt die AP mit leichten Schwankungen durchgehend in der Nähe von 0,05 und 0,1 und liefert damit konstant schlechte Ergebnisse. Völlig gegensätzlich zu den anderen Datensätzen verhält sich der Zoo-Datensatz, welcher bei 20% bereits seinen höchsten Wert erreicht, dann stark schwankt und gegen Ende hin niedrige Werte annimmt. Auch der plötzliche Anstieg bei 50% ist auffällig.

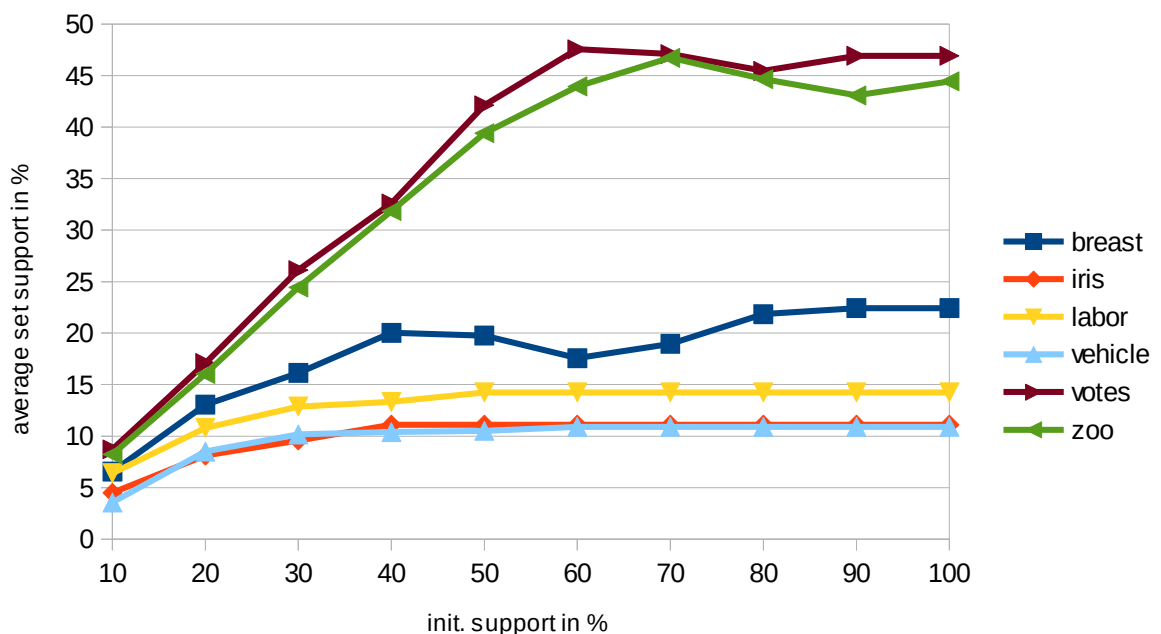


Abbildung 9: Durchschnittssupport der im IODA erzeugten Sets

Der Verlauf der AUC (Abb. 11) korreliert mit dem der AP. Steigt die AP in einem Datensatz an, so geht dies in der Regel mit einem Anstieg der AUC einher. Ebenso verhält es sich mit sinkenden Werten. Unterschiede gibt es jedoch im Grad des Anstiegs bzw. der Senkung. So führt der Schritt des init. Supports von 10% auf 20% im Iris-Datensatz zu einem kleinen Anstieg in der AP, jedoch zu einem großen Anstieg in der AUC. Beim Schritt des init. Supports von 20% auf 30% ist die Steigung in der AP dagegen groß, drückt sich in der AUC jedoch nur als geringer Anstieg aus. Als Begründung kann die unterschiedliche Bewertung der Positionen von Outliern im Ranking genannt werden. Während bei der AP erste, sehr früh erkannte Outlier bereits zu hohen Werten führen, hat die Bewertung durch die AUC einen gleichmäßigeren Charakter. Hier können schlechte Positionen der zuerst erkannten Outlier

leichter durch frühe Erkennung der letzten Outlier ausgeglichen werden. Bei der Bewertung der AP ist zu bedenken, dass der Anteil der Outlier in den Datensätzen jeweils bei etwa 5% liegt. Eine zufällige Auswahl von Instanzen liegt damit bei einer AP von etwa 0,05. Ein Wert von 0,2 bedeutet daher bereits eine Überrepräsentation von Outliern, um Faktor 4.

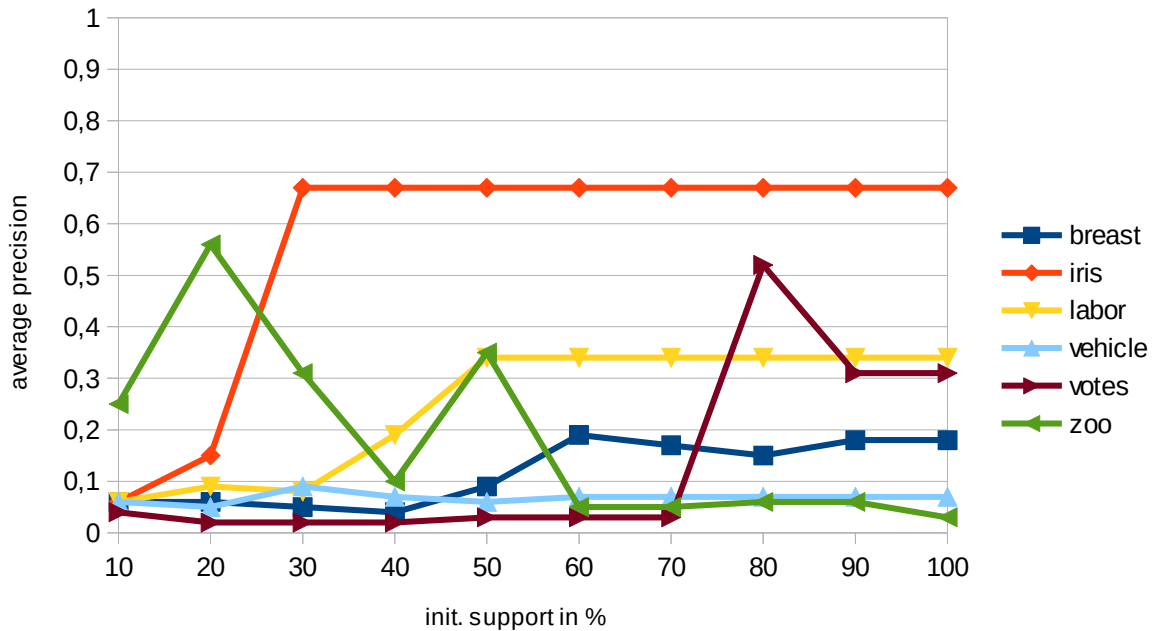


Abbildung 10: Average Precision in Abhängigkeit des init. Supports

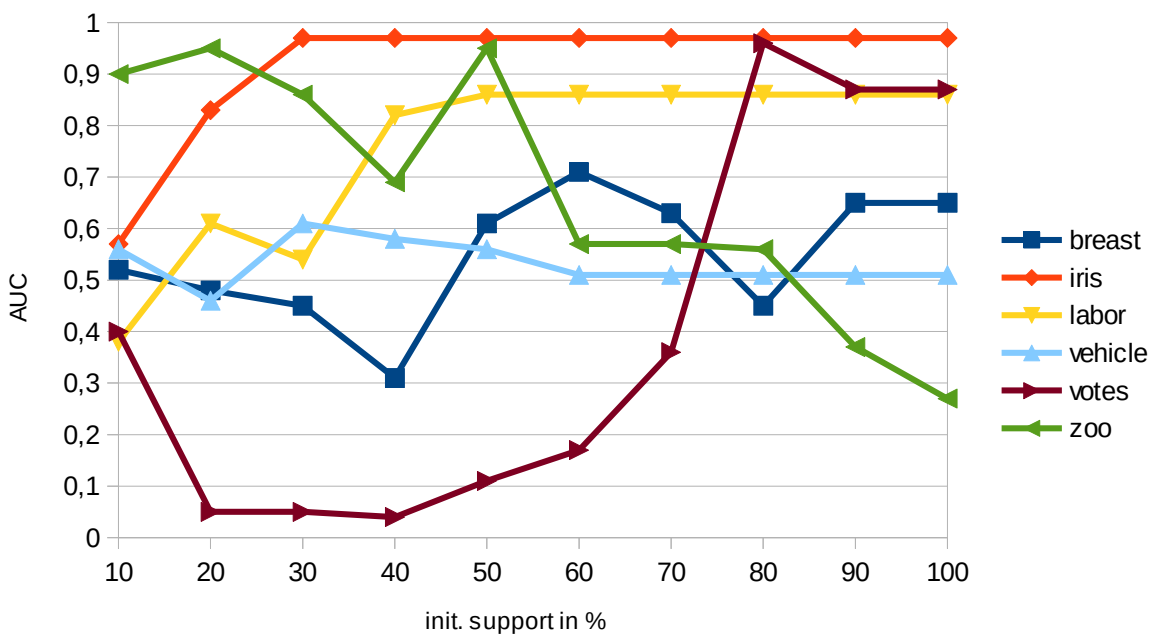


Abbildung 11: Area Under ROC Curve (AUC) in Abhängigkeit des init. Supports

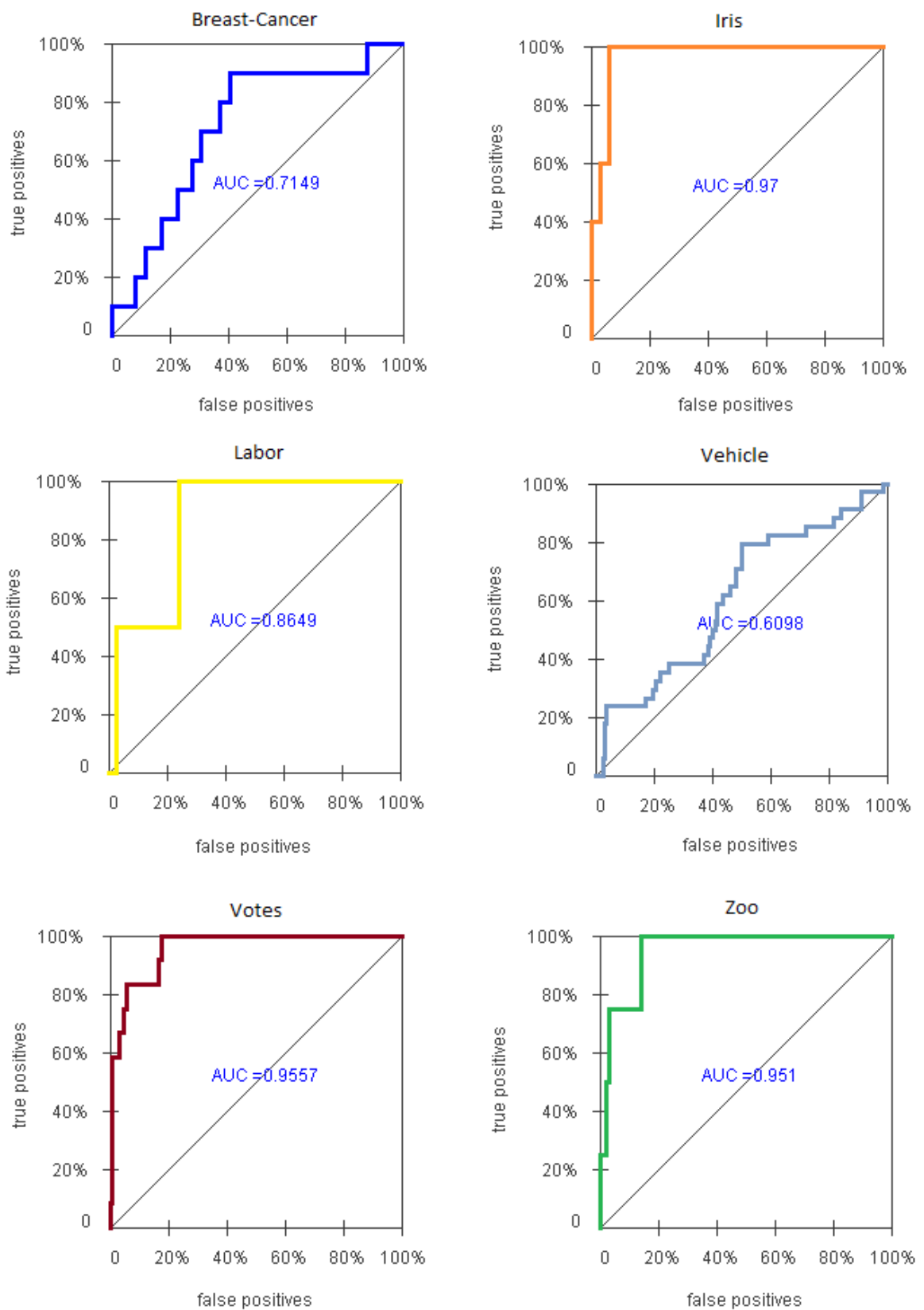


Abbildung 12: ROC-Kurven des jeweils besten Ergebnisses der verwendeten Datensätze

4.2.2 Qualitative Ergebnisse

Für die qualitative Untersuchung wurde jeweils der initiale Support für einen Datensatz verwendet, welcher die besten Ergebnisse bezüglich AP und AUC liefert. Lieferten mehrere Supportwerte identische Ergebnisse, so wurde der jeweils niedrigste Support ausgewählt. Für diese Ergebnisse werden pro Datensatz stichprobenartig die besten 4 Instanzen und ihre 5 seltensten Attributsets aufgelistet. Besonders interessant sind hierbei Sets, die einen Support von weniger als 5% aufweisen. Bei einem Outlieranteil von etwa 5% in den Datensätzen, stellen diese möglicherweise Eigenschaften dar, die ausschließlich von Outliern geteilt werden. Die identifizierten Outlier werden in folgender Form angegeben:

[Name] <[Anz. seltener Sets], [durchschnittl. Setsupport]>	
[ATTRIBUTESET 1]	[SUPPORT ATTRIBUTESET 1]
[ATTRIBUTESET 2]	[SUPPORT ATTRIBUTESET 2]
[ATTRIBUTESET 3]	[SUPPORT ATTRIBUTESET 3]
[ATTRIBUTESET 4]	[SUPPORT ATTRIBUTESET 4]
[ATTRIBUTESET 5]	[SUPPORT ATTRIBUTESET 5]

Tabelle 6: Struktur der Darstellung von identifizierten Outliern

Sämtliche Angaben des Supports sind in prozentualer Form, um Vergleichbarkeit zwischen unterschiedlichen Datensätzen zu ermöglichen. Der durchschnittliche Setsupport eines Outliers bezieht sich dabei auf alle durch die Instanz abgedeckten, seltenen Sets. Für die Interpretation der qualitativen Ergebnisse ist es außerdem wichtig, genauere Kenntnis über die Semantik der Datensätze zu haben. In der folgenden Tabelle sind daher zusätzliche Informationen bezüglich Instanzen und verwendeter Outlierklassen aufgeführt:

Datensatz	Instanzen	Outlierklasse
Breast-Cancer	an Brustkrebs erkrankte Patienten	Patienten mit wiederkehrendem Brustkrebs
Iris	Pflanzen der Gattung Schwertlilie	Schwertlilien der Unterart „Iris-Virginica“
Labor	Tarifverträge	schlechte Tarifverträge
Vehicle	Silhouettenmaße von Fahrzeugen	Silhouettenmaße von Vans
Votes	Politiker im US-Amerikanischen Kongress	Republikaner
Zoo	Tiere	Amphibien

Tabelle 7: weitere Informationen zu den verwendeten Datensätze

Betrachtet man die in Tabelle 8 aufgeführten Ergebnisse für den Breast-Cancer-Datensatz, so sticht das Attribut „inv-nodes“ hervor. Dieses Attribut gibt an wie viele Lymphknoten metastasierenden Brustkrebs enthalten. Bei allen identifizierten Instanzen erscheint das Attribut in den 5 seltensten Sets, mit einem Supportwert unter der 5%-Schwelle. Mit Attributwerten von 3 bis 17, liegen diese über dem Durchschnitt. Weiterhin tauchen die Attribute „tumor-size“, „node-caps“ und „irradiat“ in allen 4 Outliern auf. Während die Werte für das Attribute „tumor-size“ entweder besonders klein oder besonders groß sind, ist der Wert von „node-caps“ und „irradiat“ für alle aufgelisteten Instanzen mit „yes“ identisch. Das Attribut „node-caps“ gibt dabei an, ob der vorhandene Brustkrebs in einen Lymphknoten gestreut hat. Das Auftreten dieser beiden Eigenschaften, zusammen mit einem hohe „inv-nodes“-Wert und einer ungewöhnlichen Tumorgroße, scheinen also Indiz für einen Outlier zu sein. Setzt man diese Eigenschaften in Bezug zur verwendeten Outlierklasse, welche Patienten mit wiederkehrendem Brustkrebs enthält, so erscheint dies einleuchtend. Bei Patienten mit einer höheren Anzahl an mit Krebs infizierten Lymphknoten, erscheint es naheliegend dass Krebs häufiger erneut auftritt. Gleiches gilt für die vorhandene Streuung in Lymphknoten.

BC#1 <9, 25.91>		BC#2 <9, 25.8>	
{inv-nodes=6-8}	3.79	{inv-nodes=9-11}	1.90
{age=30-39}	10.90	{tumor-size=35-39}	6.64
{tumor-size=15-19}	12.32	{breast-quad=right_up}	9.95
{node-caps=yes}	13.74	{node-caps=yes}	13.74
{irradiat=yes}	18.96	{irradiat=yes}	18.96
BC#3 <9, 26.27>		BC#4 <9, 27.54>	
{tumor-size=40-44}	7.58	{inv-nodes=6-8}	3.79
{inv-nodes=3-5}	9.48	{age=30-39}	10.90
{node-caps=yes}	13.74	{node-caps=yes}	13.74
{irradiat=yes}	18.96	{tumor-size=25-29}	17.54
{deg-malig=3}	21.33	{irradiat=yes}	18.96

Tabelle 8: Top 4 Outlier im Breast-Cancer-Datensatz, (init. Support=0,6)

Besonders interessant ist die Verwendung einer Behandlungsmethode mit Röntgenstrahlen („irradiat=yes“). Hier ist einerseits möglich, dass die Therapie häufiger zu einer Wiederkehr von Krebszellen führt. Andererseits könnte es aber auch sein, dass Patienten mit wiederkehrendem Krebs häufiger mit einer Röntgentherapie behandelt werden. Was die Größe der Tumore angeht, so ließe sich hineininterpretieren, dass überdurchschnittlich große Tumore öfter wiederkehren, während sehr kleine Tumore möglicherweise schwierig zu detektieren sind und daher ebenfalls wiederkehren. Um zu überprüfen inwieweit derartige Interpretationen der Realität entsprechen, wird jedoch medizinisches Fachwissen benötigt.

Die in Tabelle 9 aufgelisteten Outlier des Iris-Datensatzes weisen je nur vier seltene Attribute auf, da der gesamte Datensatz selbst nur 4 Attribute besitzt. Kombinationen von Attributen werden für den verwendeten initialen Support nicht erzeugt. Außerdem sind erst ab Position 61 im Ranking Instanzen mit 3 oder weniger seltenen Attributen zu finden, weshalb hier in erster Linie der Durchschnittssupport der zugehörigen Sets ausschlaggebend für das Ranking der besten Instanzen ist. Für alle Outlier taucht ein besonders seltener Wert für das Attribut „petallength“ auf, der unter einem Support von 5% liegt. Dabei liegt der Attributwert für IRIS#1 und IRIS#2 im sehr hohen, für IRIS#3 und IRIS#4 eher im niedrigen mittleren Bereich. Weiterhin weisen IRIS#1 und IRIS#2 zusätzlich besonders seltene Werte für das Attribut „petalwidth“ auf, die ebenfalls beide im hohen Bereich liegen. Für IRIS#3 und IRIS#4 sind die Werte nicht ganz so selten, liegen aber auch hier wieder im niedrigen mittleren Bereich.

IRIS#1 <4, 6.43>		IRIS#2 <4, 6.67>	
{petalwidth='(2.26-inf)'} {petallength='(5.72-6.31]}'	0,95 2,86	{petallength='(5.13-5.72]}' {petalwidth='(1.78-2.02]}'	0,95 2,86
{sepallength='(6.1-6.46]}' {sepalwidth='(3.2-3.44]}'	8,57 13,33	{sepallength='(6.1-6.46]}' {sepalwidth='(2.72-2.96]}'	8,57 14,29
IRIS#3 <4, 9.29>		IRIS#4 <4, 9.29>	
{petallength='(2.77-3.36]}' {petalwidth='(0.82-1.06]}'	2,86 6,67	{petallength='(2.77-3.36]}' {petalwidth='(0.82-1.06]}'	2,86 6,67
{sepalwidth='(2.24-2.48]}' {sepallength='(4.66-5.02]}'	6,67 20,95	{sepalwidth='(2.24-2.48]}' {sepallength='(4.66-5.02]}'	6,67 20,95

Tabelle 9: Top 4 Outlier im Iris-Datensatz,(init. Support=0,3)

Als Eigenschaften für Outlier ließen sich somit entweder gleichzeitig hohe Werte für „petallength“ und „petalwidth“ oder niedrige, mittlere Werte für die beiden Attribute ausmachen. Besonders ausschlaggebend scheinen dabei die Werte für das Attribut „petallength“ zu sein. Dies passt zu den Beobachtungen im Datensatz. Die als Outlierklasse verwendete Klasse der „iris-virginica“ zeichnet sich tatsächlich durch hohe Werte in den Attributen „petallength“ und „petalwidth“ aus. Insgesamt gibt es aber auch nur sehr wenige Instanzen im Datensatz, mit niedrig-mittleren Werten für die beiden Attribute, was die Identifikation als Outlier von IRIS#3 und IRIS#4 erklärt.

Tabelle 10 zeigt die besten Outlier des Labor-Datensatzes, bei einem Support von 0,5. Dabei hebt sich Outlier L#1 mit 13 seltenen Sets von den übrigen Outliern ab, weist jedoch einen deutlich höheren Durchschnittssupport auf. Was die besonders seltenen Attribute, mit einem Support unter 5% betrifft, sind diese für die aufgelisteten Instanzen sehr unterschiedlich. Im Allgemeinen zeichnet sich jedoch ab, dass „wage-increase-first-year“ und „wage-increase-second-year“ wichtige Attribute für die Bewertung als Outlier sind. Die Werte von „wage-increase-first-year“ und „wage-increase-second-year“ liegen für Outlier L#1 und L#3 im mittleren, für L#4 im hohen Bereich. Was die Werte dieser Attribute betrifft, zeichnet sich daher kein klares Bild ab. Es erscheint jedoch naheliegend, dass die Gehaltserhöhung im Allgemeinen ein gutes Kriterium für die Bewertung eines Arbeitsvertrages ist. Eine Ausnahme hiervon bildet Outlier L#2, für den vor allem fehlende Pension, fehlende Zuschüsse für die Krankenkasse und fehlende Unterstützung für Langzeiterkrankung ausschlaggebend sind. Hinzu kommen fehlende Leistungen für Zahnbehandlungen und eine geringe Anzahl von gesetzlichen Feiertagen. Alle 5 Eigenschaften sind nachvollziehbare Gründe für die schlechte Bewertung eines Tarifvertrags. Während L#1, L#2 und L#3 tatsächlich Beispiele für schlechte Tarifverträge sind, erscheint L#4 eher wegen außergewöhnlich guter Konditionen so hoch im Ranking.

L#1 <13, 30.97>		L#2 <12, 20.95>	
{working-hours='(36.1-37.4]}'	7,69	{pension=none}	2,56
{education-allowance=no}	10,26	{contribution-to-health-plan=none}	2,56
{wage-increase-first-year='(4.5-5]}'	15,38	{longterm-disability-assistance=no}	5,13
{vacation=below_average}	23,08	{contribution-to-dental-plan=none}	7,69
{wage-increase-second-year='(3.5-4]}'	30,77	{statutory-holidays='(9.6-10.2]}'	10,26
L#3 <12, 22.64>		L#4 <12, 22.64>	
{cost-of-living-adjustment=tc}	5,13	{wage-increase-third-year='(3.86-4.17]}'	2,56
{contribution-to-dental-plan=none}	7,69	{shift-differential='(12.5-15]}'	2,56
{wage-increase-first-year='(3.5-4]}'	12,82	{wage-increase-second-year='(5.5-6]}'	5,13
{wage-increase-second-year='(4.5-5]}'	12,82	{statutory-holidays='(-inf-9.6]}'	5,13
{wage-increase-third-year='(4.79-inf)'}	15,38	{wage-increase-first-year='(5.5-6]}'	10,26

Tabelle 10: Top 4 Outlier im Labor-Datensatz, (init. Support=0,5)

Die Ergebnisse für den Vehicle-Datensatz in Tabelle 11 zeigen mit „skewness about_minor='(15.4-17.6]“ nur einen einzigen Attributwert, mit einem Support unter 5%, welcher auch nur bei VE#1 vorkommt. Das Attribut „hollows ratio“ taucht dagegen bei allen aufgelisteten Outliern auf und liegt mit Werten zwischen 187 und 196 im niedrigen, bis mittleren Bereich. Als weiteres Kriterium für Outlier kann das Attribut „max.length rectangularity“ gelten, das ebenfalls bei allen Outlier erscheint. Dabei liegen die Werte mit 125 bis 139 im niedrigen Wertebereich. Als Eigenschaften für Outlier könnten hier also niedrige bis mittlere Werte für „hollows ratio“, bei gleichzeitig niedrigen Werten für „max.length rectangularity“ ausgemacht werden.

Bei Betrachtung der Werte im Datensatz, erscheint jedoch keines der gefunden Attribute als guter Identifikator für die verwendete Outlierklasse „Van“. Als Grund dafür kann die allgemein schlechte Identifizierung von Outliern in diesem Datensatz angegeben werden.

VE#1 <23, 16.81>		VE#2 <23, 18.43>	
{skewness about_minor='(15.4-17.6]}'	2,64	{hollows ratio='(187-190]}'	5,87
{hollows ratio='(187-190]}'	5,87	{kurtosis about_minor='(182-185]}'	9,99
{kurtosis about_minor='(182-185]}'	9,99	{distance circularity='(61.6-68.8]}'	12,04
{max.length rectangularity='(132-139]}'	12,19	{scaled variance_major='(149-168]}'	13,51
{distance circularity='(76-83.2]}'	12,92	{compactness='(82.2-86.8]}'	14,98
VE#3 <23, 18.47>		VE#4 <23, 18.54>	
{hollows ratio='(190-193]}'	7,64	{circularity='(38.2-40.8]}'	8,37
{kurtosis about_major='(20.5-24.6]}'	10,72	{max.length rectangularity='(132-139]}'	12,19
{distance circularity='(76-83.2]}'	12,92	{kurtosis about_minor='(191-194]}'	12,63
{radius of gyration='(140.8-156.7]}'	12,92	{radius of gyration='(140.8-156.7]}'	12,92
{max.length rectangularity='(125-132]}'	13,36	{hollows ratio='(193-196]}'	14,68

Tabelle 11: Top 4 Outlier im Vehicle-Datensatz (init. Support=0,3)

Für den Votes-Datensatz in Tabelle 12 zeigen die Top 4 Outlier ein sehr einheitliches Bild. So weisen alle die selben Werte für die Attribute „physician-fee-freeze“, „adoption-of-the-br“, „education-spending“ und „aid-to-nicaraguan-contras“ auf. Weiterhin haben die ersten drei Instanzen den selben Wert für „export-administration-act-sa“, die letzten beiden den selben Wert für „el-salvador-aid“. Outlier ließen sich hier also dadurch charakterisieren, dass sie für „physician-fee-freeze“ und „education-spending“ gestimmt haben, gegen „adoption-of-the-br“ und „aid-to-nicaraguan-contras“ und zusätzlich entweder gegen „export-administration-act-sa“ oder für „el-salvador-aid“. Tatsächlich decken sich die Werte mit dem mehrheitlichen Abstimmungsverhalten der als Outlierklasse verwendeten Republikaner. So haben Republikaner jeweils mit großer Mehrheit für „physician-fee-freeze“ und „education-spending“ und gegen „adoption-of-the-br“, „aid-to-nicaraguan-contras“ und „el-salvador-aid“ gestimmt. Nur bei „export-administration-act-sa“ hat nur etwa ein Drittel der Republikaner für und zwei Drittel dagegen gestimmt.

Die Ergebnisse des Zoo-Datensatzes in Tabelle 13 heben sich dadurch hervor, dass die seltensten Sets der Outlier alle zwei Attribute enthalten. Das Attribut „legs“ taucht dabei in jeder der Kombinationen auf und hat bei ZOO#1, ZOO#2 und ZOO#4 jeweils den Wert 4, bei Outlier ZOO#3 Wert 0. Daraus lässt sich schließen, dass es für Tiere im Datensatz mit den Eigenschaften „hair=false“, „milk=false“, „eggs=true“, ungewöhnlich ist vier Beine zu besitzen. Üblicherweise beschreiben die ersten drei Attribute in Kombination mit „aquatic=true“ die Tiergruppe der Fische. Durch den zusätzlichen Besitz von vier Beinen, können ZOO#1 und ZOO#4 aber klar als Outlierklasse definierten Tiergruppe der Amphibien zugeordnet werden. ZOO#2 scheint ein amphibienähnliches Tier zu sein, das jedoch nicht im Wasser lebt. Gleiches gilt für Outlier ZOO#3, der durch fehlende Bein in Kombination mit der

Eigenschaft Raubtier zu sein wohl eine Schlange zugeordnet werden kann. Für beide kommt daher die Tiergruppe der Reptilien in Frage, die mit nur 5 Instanzen im Datensatz ebenfalls selten ist.

VO#1 <16, 29.57>		VO#2 <16, 29.93>	
{export-administration-act-sa=n}	5,38	{export-administration-act-sa=n}	5,38
{physician-fee-freeze=y}	9,32	{physician-fee-freeze=y}	9,32
{adoption-of-the-budget-resolution=n}	14,70	{adoption-of-the-br=n}	14,70
{education-spending=y}	15,77	{education-spending=y}	15,77
{aid-to-nicaraguan-contras=n}	20,07	{aid-to-nicaraguan-contras=n}	20,07
VO#3 <16, 31.95>		VO#4 <16, 33.58>	
{export-administration-act-sa=n}	5,38	{physician-fee-freeze=y}	9,32
{physician-fee-freeze=y}	9,32	{adoption-of-the-br=n}	14,70
{adoption-of-the-budget-resolution=n}	14,70	{education-spending=y}	15,77
{education-spending=y}	15,77	{aid-to-nicaraguan-contras=n}	20,07
{aid-to-nicaraguan-contras=n}	20,07	{el-salvador-aid=y}	24,01

Tabelle 12: Top 4 Outlier im Votes-Datensatz,(init. Support=0,8)

ZOO#1 <225, 17.68>		ZOO#2 <221, 17.22>	
{hair=false, legs=4}	7	{hair=false, legs=4}	7
{milk=false, legs=4}	7	{milk=false, legs=4}	7
{aquatic=true, legs=4}	7	{eggs=true, legs=4}	8
{eggs=true, legs=4}	8	{legs=4, catsize=false}	12
{legs=4, catsize=false}	12	{eggs=true, toothed=true}	20
ZOO#3 <220, 17.20>		ZOO#4 <169, 17.41>	
{aquatic=false, legs=0}	5	{hair=false, legs=4}	7
{breathes=true, legs=0}	7	{milk=false, legs=4}	7
{fins=false, legs=0}	7	{aquatic=true, legs=4}	7
{legs=0, catsize=false}	16	{legs=4, tail=false}	7
{predator=true, legs=0}	17	{eggs=true, legs=4}	8

Tabelle 13: Top 4 Outlier im Zoo-Datensatz,(init. Support=0,2)

4.3 Schlussfolgerung

Auch wenn die Genauigkeit der Outliererkennung bei dieser Arbeit nicht im Vordergrund steht, konnte gezeigt werden, dass für die meisten der verwendeten Datensätze ein initialer Support für den IODA existiert, der gute Ergebnisse liefert. Durch unterschiedliche Strukturen in den Daten und Unterschiede bei den erzeugten Attributsets, gestaltet sich die Bestimmung eines guten initialen Supports jedoch schwierig. Dabei scheint auch eine erste statistische Betrachtung der erzeugten Sets keinen Schluss auf einen guten Supportwert zuzulassen.

Die Verwendung von seltenen Attributsets als Erklärungen für Outlier, ist dagegen ein gutes Mittel, um die Interpretierbarkeit der Outlier Detection zu verbessern. Neben Anzahl und durchschnittlichem Support der zugehörigen Sets, die für das Ranking der Instanzen verwendet werden, zeigen die seltensten Sets Besonderheiten eines Outliers auf. Durch Betrachtung dieser Sets, ist es einem Anwender möglich ungewöhnliche Eigenschaften eines Outliers nachzuvollziehen. Durch den Vergleich mehrerer Outlier und ihrer seltenen Attributsets untereinander, können außerdem allgemeinere Kriterien ausgemacht werden, die Aussagen über die Eigenschaften von Outliern zulassen. Diese Art der Präsentation von Outliern geht über die übliche Darstellung hinaus, bei der dem Nutzer in der Regel nur ein Ranking von Outliern zur Verfügung steht und ermöglicht eine weitergehende Interpretation der Ergebnisse.

Auffällig ist jedoch die häufige Auswahl von Attributsets mit nur einem Attribut. Während im niedrigen Supportbereich im Schnitt längere Attributsets entstehen, liegen die Ergebnisse mit guter Identifikation oft in einem Supportbereich, bei dem in erster Linie Sets der Länge 1 entstehen. Ein Grund dafür kann die identische Bewertung von Sets mit unterschiedlicher Länge sein. Möglicherweise entstehen bessere Ergebnisse in niedrigeren Supportbereichen, wenn längere Sets stärker gewichtet werden. So könnten Sets von größerer Länge angegeben werden, die eventuell noch besser interpretierbare Begründungen für Outlier liefern. Aufgrund der zeitlichen Begrenzung der Arbeit konnte dies jedoch leider nicht mehr überprüft werden.

5. Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten präsentiert, die entweder ebenfalls die Zielsetzung einer interpretierbaren Outlier Detection haben oder ähnliche Ansätze zur Identifikation von Outliern verfolgen, wie der vorgestellte IODA.

Die Arbeiten [14], [17] und [18] nutzen ebenfalls Methoden aus dem Frequent Itemset Mining, um Outlier zu identifizieren. In [16] werden Regressionslerner in Verbindung mit einem gewichtete Distanzmaß verwendet. In [19] werden dagegen Methoden aus der Informationstheorie verwendet, um Outlier zu identifizieren und interessante Informationen über Outlier zu generieren.

5.1 Finding Intensional Knowledge of Distance-Based Outliers

Die Arbeit von Edwin M. Knorr und Raymond T. Ng [14] ist die vermutlich erste Arbeit, welche sich mit der Generierung von zusätzlichen Informationen für Outlier auseinandersetzt. Dort wird nach der geringstmöglichen Attributmenge gesucht, die erklärt warum eine Instanz als Outlier gilt. Dafür werden zunächst drei Arten von Outliern definiert: nicht-triviale, starke und schwache Outlier. Nicht-triviale Outlier zeichnen sich dadurch aus, dass sie bezüglich einer Attributmenge als Outlier gelten, ohne dass sie Outlier in einer ihrer Untermengen sind. Starke Outlier sind dagegen Outlier in einer Attributmenge, in deren Untermenge allgemein keine Outlier mehr existieren. Schwache Outlier sind alle nicht-trivialen Outlier, die keine starken Outlier sind.

Player Name	Power-play Goals	Short-handed Goals	Game-winning Goals	Game-tying Goals	Games Played
MARIO LEMIEUX	31	8	8	0	70
JAROMIR JAGR	20	1	12	1	82
JOHN LECLAIR	19	0	10	2	82
ROD BRIND'AMOUR	4	4	5	4	82

Abbildung 13: Attributwerte von Outliern in einem Datensatz über NHL-Spieler
Quelle: [14], Abb. 1

Für die Identifikation von starken und schwachen Outliern in einer gegebenen Attributmenge, werden zwei Algorithmen verwendet. Für Mengen mit Kardinalität $k \geq 5$ ist der NL-Algorithmus [15] am performantesten. Hier werden in einer verschachtelten Schleife blockweise die Distanzen zwischen Outliern bestimmt. Für Attributmengen mit $k \leq 5$ ist der CELL-Algorithmus [15] günstiger, bei dem Tupel jeweils einer sogenannten Zelle zugeordnet werden. Dafür wird die Attributmenge in Zellen eingeteilt, die jeweils die Länge $\frac{D}{2\sqrt{k}}$ in jeder Dimension haben, wobei D den Radius der lokalen Nachbarschaft bezeichnet. Zellen die zu viele Tupel beinhalten oder die an Zellen mit zu vielen Tupeln angrenzen werden verworfen. Für alle anderen Zellen werden die die Distanzen der Tupel exakt berechnet.

Unter Verwendung dieser beiden Algorithmen, werden vier unterschiedliche Ansätze vorgestellt, die sich darin unterscheiden wie die jeweiligen Attributmengen abgearbeitet werden. Beim UpLattice-Algorithmus (ULA) werden Attributmengen, wie beim Frequent

Itemset Mining, in einem Bottom-Up-Verfahren jeweils um eine Dimension erweitert und entsprechende Outlier identifiziert. Beim JumpLattice-Algorithmus (JLA) geschieht dies auf ähnliche Weise, jedoch wird erst ab einer vorgegebenen Dimension begonnen und nur bei Identifikation eines Outliers in der Attributmenge, werden auch Untermengen mit geringerer Dimension untersucht. Ein optimaler Wert, bei welcher Dimension begonnen werden sollte, kann jedoch nicht zuverlässig angegeben werden. Eine heuristische Untersuchung zeigt aber, dass sich die zusätzlichen Kosten bis zur 3. Dimension in Grenzen halten und daher ohne Bedenken dort angesetzt werden kann. Um die Performanz des JLA zu verbessern, werden zwei weitere Varianten vorgestellt. Beim JumpLattice-Path-Algorithmus (JLPA) wird der Algorithmus so angepasst, dass mehrere Attributmengen innerhalb eines Pfades im Suchraum gleichzeitig verarbeitet werden können. Im JumpLattice-Semi-Algorithmus (JLSA) werden ganze Teilbäume gruppiert und gleichzeitig abgearbeitet. JLPA und JLA zeigen dabei eine deutlich bessere Performanz als ULA und JLA. Zwischen JLPA und JLA sind jedoch nur geringe Unterschiede bezüglich Performanz auszumachen. Als zusätzliche Informationen über die Outlier, dienen schließlich die Attributmengen, in denen sie als starke oder schwache Outlier identifiziert wurden. Eine mögliche Darstellung der erzeugten Ergebnisse ist in Abb. 14 zu sehen.

```

MARIO LEMIEUX:
  (i) An outlier in the 1-D space of Power-play goals
  (ii) An outlier in the 2-D space of Short-handed goals and
      Game-winning goals
      (No player is exceptional on Short-handed goals alone;
       No player is exceptional on Game-winning goals alone.)
ROD BRIND'AMOUR:
  (i) An outlier in the 1-D space of Game-tying goals
JAROMIR JAGR:
  (i) An outlier in the 2-D space of Short-handed goals and
      Game-winning goals
      (No player is exceptional on Short-handed goals alone;
       No player is exceptional on Game-winning goals alone.)
  (ii) An outlier in the 2-D space of Power-play goals and
      Game-winning goals
      (But for Power-play goals alone, the current player is
       dominated by another and is not exceptional.)
JOHN LECLAIR:
  (i) An outlier in the 2-D space of Game-winning goals and
      Game-tying goals
      (But for Game-tying goals alone, the current player is
       dominated by another and is not exceptional.)

```

Abbildung 14: erzeugte Informationen über gefundene Outlier
Quelle: [14], Abb. 2

5.2 Attribute-wise Learning for Scoring Outliers

In der Arbeit von Heiko Paulheim und Robert Meusel [16] wird die Identifikation von Outliern durch Aufteilung des Problems in mehrere, überwachte Regressionslerner gelöst. Dabei wird für jedes Attribut eine Vorhersagemodell erstellt, das auf den Werten der übrigen Attribute basiert. Der tatsächliche Wert des zu bestimmenden Attributs dient dann als Label, wodurch Abweichungen berechnet werden können, die sowohl zur unterschiedlichen Gewichtung der

Attribute beitragen, als auch für das Outlierscoring verwendet werden. Dabei erhalten Attribute mit starken Mustern eine hohe und Attribute mit schwachen oder ohne erkennbare Muster niedrige bzw. gar keine Gewichtung. Dahinter steckt die Idee, dass nur dort Outlier erkannt werden können, wo auch Muster vorhanden sind. Weist ein Attribut keine Muster auf, so kann auch keine Abweichung davon bestimmt werden. Um diesen Gedanken in der Gewichtung zu berücksichtigen, wird der „root relative squared error“ (RRSE) verwendet. Der RRSE wird bei m Instanzen in einem Datensatz definiert durch:

$$R_k := \sqrt{\frac{\sum_{k=1}^m ((i_j)_k - (i'_j)_k)^2}{\sum_{k=1}^m ((i_j)_k - \bar{k})^2}} \quad (7)$$

Die Notation $(i_j)_k$ gibt dabei das k -te Attribut der j -ten Instanz an. $(i'_j)_k$ entspricht der auf dem erlernten Modell basierende Vorhersage des k -ten Attributs der j -ten Instanz. Mit der euklidischen Distanz als Grundlage, entsteht unter der Verwendung des RRSE schließlich folgendes, gewichtetes Distanzmaß:

$$o(i) = \sqrt{\frac{\sum_{k=1}^n w_k * (i_k - i'_k)^2}{\sum_{k=1}^n 1 - \min(1, R_k)}} \quad (8)$$

Dadurch erhalten Attributewerte, die stark vom erlernten Modell abweichen und ein hohe Gewichtung haben, großen Einfluss auf den Score. Attribute deren Werte schlechter bestimmt werden können als eine zufällig Auswahl ($R_k > 1$), werden dagegen nicht miteinbezogen. Gleichzeitig kann das Maß jedoch dazu führen dass redundante Outlier möglicherweise nicht erkannt werden, da diese theoretisch eine gegenseitige Vorhersage ermöglichen.

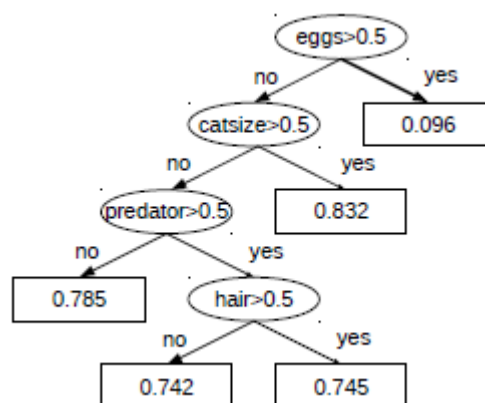


Abbildung 15: mit M5' im ALSO erstellter Entscheidungsbaum
Quelle: [16], Abb. 8 (c)

Da für jedes Attribut ein eigener Score entsteht, können die Attribute mit den höchsten Scores ausgewählt und als Erklärungen für Outlier verwendet werden. Unter Verwendung von Lernalgorithmen mit symbolischem Ansatz als Regressionslerner, können zusätzlich die erzeugten Modelle als zusätzliche Information dienen, sofern diese gut interpretierbare Modelle erzeugen. Als Beispiel hierfür wurde der M5'-Algorithmus verwendet, welcher Regressionsbäume erzeugt. Dabei entsteht jeweils ein Entscheidungsbaum pro Attribut, welcher mehr Informationen über einen Outlier bietet, als die Angabe der Attribute mit höchstem Score.

5.3 Frequent Pattern Discovery Method for Outlier Detection

In der Arbeit von Zengyou He und Shengchun Deng [17] werden häufige Itemsets für die Identifikation von Outliern verwendet. Häufige Itemsets stellen gewöhnliche Muster in einem Datensatz dar. Instanzen die nur wenige oder keine gewöhnlichen Muster aufweisen, werden als Outlier aufgefasst. Um zu bestimmen in welchem Maß eine Instanz i häufige Muster aufweist, wird der Frequent Pattern Outlier Factor (FPOF) definiert:

$$FPOF(i) := \frac{\sum_{f \subseteq i} support(f)}{|FPS|} \quad (9)$$

mit $f \subseteq i$ und $FPS :=$ Menge aller häufigen Itemsets

Ein hoher FPOF-Wert bedeutet dabei, dass eine Instanz mehr häufige Itemsets beinhaltet, wobei die Werte immer im Intervall $[0,1]$ liegen. Um Beschreibungen für einen Outlier zu liefern, werden häufige Itemsets verwendet, die nicht durch den jeweiligen Outlier abgedeckt werden. Je höher der Support des Sets, desto größer der Widerspruch zum Outlier. Außerdem wird davon ausgegangen, dass längere Itemsets bessere Beschreibungen liefern. Unter diesen beiden Annahmen entsteht das Maß der Contradictness, welches angibt wie stark ein Itemset einem Outlier widerspricht:

$$C(i, f) := (|f| - |i \cap f|) * support(f) \quad (10)$$

Nach Identifikation aller häufigen Itemsets und der Berechnung der FPOF-Werte aller Instanzen, werden schließlich die n besten Outlier und ihre k widersprüchlichsten Itemsets aufgelistet.

5.4 Discovering the Minimal Set of Unexpected Patterns

Die Arbeit von Balaji Padmanabhan und Alexander Tuzhilin [18] behandelt die Suche nach einer minimalen Menge von unerwarteten Mustern in Daten. Als unerwartet wird dabei der Widerspruch in Bezug auf eine vorherige Erwartung verstanden. Die Minimalität soll gewährleisten, dass die Menge der erzeugten Muster möglichst interessante und relevante Muster erzeugt. Die unerwarteten Muster können dabei als Outlier aufgefasst werden.

Als Input erhält der entwickelte MinZoomUR-Algorithmus (MZUR) eine Menge von Regeln der Form $Body \Rightarrow Head$, die als Erwartungen aufgefasst werden. Als Output generiert

MZUR Widerspruchsregeln, die den Erwartungen widersprechen. Als widersprüchlich zu einer Erwartung gilt eine Regel dann, wenn der Body beider auf einer statistisch relevanten Menge der Daten gilt. Außerdem muss aus der Menge beider Bodys der Head der Erwartung folgen, wobei aus dem Body der Widerspruchsregel alleine jedoch ein Widerspruch zum Head der Erwartung entsteht. Bezüglich der Minimalität gilt, dass die Menge aller minimalen Regeln nur jene enthält, aus denen alle anderen Widerspruchsregeln geschlossen werden können, ohne dass es weitere Regeln gibt, aus denen sich eine minimal Regel schließen lässt.


MZUR folgt dem Prinzip des Apriori-Algorithmus, wobei Itemsets die zu nicht-minimalen Regeln führen nicht generiert werden. Außerdem ist der Erzeugungsprozess von Regeln und Itemsets verknüpft, um die Performanz des Algorithmus zu verbessern. Dafür werden für jede Erwartung zunächst alle Heads generiert, die dem Head der Erwartung widersprechen. Kandidaten für widersprüchliche Regeln entstehen dann aus der Kombination des Bodys der Erwartung und einem der widersprüchlichen Heads. Es werden schließlich Support und Konfidenz der Kandidatenregeln bestimmt und anhand entsprechender Schwellwerte entschieden, ob diese in die Menge der minimalen unerwarteten Regeln aufgenommen werden. Neue Kandidaten für die nächste Iteration werden schließlich durch das Hinzufügen von Bedingungen, in Form von Attributwerten, generiert.

Durch Unterteilung in allgemeine und spezielle Regeln, die vom gleichen Kandidaten generiert werden, können sämtliche nicht-minimalen Regeln eliminiert werden. Insgesamt werden vier auf diesem Konzept basierende Bedingungen präsentiert, die die Minimalität der Regeln gewährleisten. Am Ende wird die Menge aller minimalen Regeln ausgegeben, die widersprüchliche Zusammenhänge im Datensatz darstellen.

5.5 Local Outlier Detection with Interpretation

Die Arbeit von Xuan Hong Dang u.a. [19] behandelt die interpretierbare Identifikation von lokalen Outliern. Dafür wurde der LODI-Algorithmus entwickelt, welcher Mengen von Nachbarinstanzen auswählt und nach Untermengen sucht, in denen sich Outlier stark von ihren Nachbarn unterscheiden. Anstatt jedoch, wie in vielen anderen Ansätzen, die k nächsten Nachbarn zu betrachten, wird nur eine untere Grenze für benachbarte, normale Instanzen gesetzt. Die lokale quadratischen Entropie, welche eine Variation des aus der Informationstheorie bekannten Entropiemaßes darstellt, wird als Maß für die Ähnlichkeit von Objekten in einer Menge von Nachbarn verwendet. Niedrige Entropiewerte deuten dabei auf große Ähnlichkeiten zwischen den Objekten hin, während hohe Entropiewerte auf Outlier hinweisen.

Für die Bestimmung von Nachbarmengen wird zunächst eine initiale Anzahl von nächsten Nachbarn festgelegt. Es wird dann durch einen heuristischen Ansatz versucht jene Untermengen zu identifizieren, die eine möglichst geringe Entropie aufweisen. Dafür werden Objekte aus der Menge entfernt und untersucht, wie sehr sich die Entropie verringert. Es entsteht ein Ranking nach Entropiewerten, durch das interessante Objekte von uninteressanten getrennt werden können. Es werden schließlich Varianz- und Distanzmaße für Objekte und zugehörige Nachbarmengen formuliert, die unter Verwendung von Matrix-



Eigendekomposition gelöst werden können. Die Koeffizienten im Eigenvektor stellen schließlich die Gewichtung der Attribute dar. Dadurch können jene Attribute mit den höchsten Koeffizienten als ausschlaggebende Faktoren bestimmt werden. Durch Präsentation der am stärksten gewichteten Attribute und ihrer Gewichte, werden dem Nutzer Informationen darüber vermittelt, in welchen Bereichen ein Outlier ungewöhnliche Eigenschaften aufweist.

6. Zusammenfassung und Ausblick

In dieser Arbeit wurde der IODA für die Identifikation von Outliern mit interpretierbaren Ergebnissen entwickelt. Unter Verwendung von Methoden aus dem Frequent Pattern Mining, wird dabei in einem unüberwachten Verfahren ein Modell für Punkt-Outlier erstellt, anhand dessen ein Ranking von Instanzen konstruiert wird. Darüber hinaus werden die jeweils seltensten Attributkombinationen der identifizierten Outlier präsentiert, um einem Anwender ihre Eigenschaften und Besonderheiten zu veranschaulichen.

Die durchgeführten Experimente zeigen, dass für die Mehrheit der verwendeten Datensätze Werte gefunden werden können, die zu einer guten Identifikation von Outliern führen. Die Bestimmung dieser Werte gestaltet sich aufgrund unterschiedlicher Beschaffenheit der Datensätze jedoch schwierig. Die Auswahl von seltenen Eigenschaften zeigt sich aber als gute Möglichkeit, um die Charakteristiken von Outliern zu veranschaulichen und einem Anwender zusätzliches Wissen über die Daten zu vermitteln. Dieses ermöglicht wiederum weitergehende Interpretationen, die mit entsprechendem Fachwissen verifiziert werden können. Die Untersuchungen haben außerdem gezeigt, dass sich die Interpretationsmöglichkeiten oft mit den Beobachtungen im Datensatz decken, sofern eine gute Identifikation der Outlier stattgefunden hat. Maßgeblich für die Qualität der Interpretation ist dabei ein Verständnis der in einem Datensatz enthaltenen Attribute und Instanzen. Nominale Attribute sind dabei oft leichter nachvollziehbar als numerische.

Zur Verbesserung der Identifikation von Outliern im IODA, könnte eine genauere Betrachtung der Datensätze und der für sie erzeugten Outliermodelle von Vorteil sein, um die Bestimmung eines guten initialen Supports zu erleichtern. Zusätzlich könnte eine unterschiedliche Gewichtung von Attributsets mit unterschiedlicher Länge vorgenommen werden. Dadurch könnten ebenso mehr längere Sets als relevant identifiziert werden, die Zusammenhänge im Datensatz möglicherweise besser wiedergeben. Weiterhin könnten jene seltenen Attributsets von besonderem Interesse sein, deren Untermengen sehr häufig sind. Führt die Vereinigung von Itemsets mit einem sehr hohen Support zu einem Itemset mit einem sehr niedrigen Support, so ließe sich dies als widersprüchlich auslegen. Das Verhältnis vom Support der Untermengen zum Support des Sets, könnte dabei als Maß der Widersprüchlichkeit verwendet werden. Eine Auflistung der in diesem Sinne widersprüchlichsten Itemsets und ihrer häufigsten Untermenge, könnte einem Anwender zusätzliches Wissen über den Datensatz vermitteln.

7. Literaturverzeichnis

- [1] Frawley, William J., Gregory Piatetsky-Shapiro, and Christopher J. Matheus. "Knowledge discovery in databases: An overview." *AI magazine* 13.3 (1992): 57.
- [2] Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From data mining to knowledge discovery in databases." *AI magazine* 17.3 (1996): 37.
- [3] Kumar, Vipin. "Parallel and distributed computing for cybersecurity." *Distributed Systems Online, IEEE* 6.10 (2005).
- [4] Spence, Clay, Lucas Parra, and Paul Sajda. "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model." *Mathematical Methods in Biomedical Image Analysis, 2001. MMBIA 2001. IEEE Workshop on. IEEE, 2001.*
- [5] Aleskerov, Emin, Bernd Freisleben, and Bharat Rao. "Cardwatch: A neural network based database mining system for credit card fraud detection." *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997. IEEE, 1997.*
- [6] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." *ACM computing surveys (CSUR)* 41.3 (2009): 15.
- [7] Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." *Acm sigmod record. Vol. 22. No. 2. ACM, 1993.*
- [8] Goethals, Bart. "Survey on frequent pattern mining." *Univ. of Helsinki (2003).*
- [9] Mannila, Heikki, and Hannu Toivonen. "Levelwise search and borders of theories in knowledge discovery." *Data mining and knowledge discovery* 1.3 (1997): 241-258.
- [10] Hall, Mark, et al. "The WEKA data mining software: an update." *ACM SIGKDD explorations newsletter* 11.1 (2009): 10-18.
- [11] Asuncion, Arthur, and David Newman. "UCI machine learning repository." (2007).
- [12] Aggarwal, Charu C., and Philip S. Yu. "Outlier detection for high dimensional data." *ACM Sigmod Record. Vol. 30. No. 2. ACM, 2001.*
- [13] Kirkby, Richard, Eibe Frank, and Peter Reutemann. "Weka explorer user guide for version 3-5-8." *University of Waikato (2007).*
- [14] Knorr, Edwin M., and Raymond T. Ng. "Finding intensional knowledge of distance-based outliers." *VLDB. Vol. 99. 1999.*
- [15] Knox, Edwin M., and Raymond T. Ng. "Algorithms for mining distancebased outliers in large datasets." *Proceedings of the International Conference on Very Large Data Bases. 1998.*
- [16] Paulheim, Heiko, and Robert Meusel. "A decomposition of the outlier detection problem into a set of supervised learning problems." *Machine Learning* 100.2-3 (2015): 509-531.

[17] He, Zengyou, et al. "A frequent pattern discovery method for outlier detection." International Conference on Web-Age Information Management. Springer Berlin Heidelberg, 2004.

[18] Padmanabhan, Balaji, and Alexander Tuzhilin. "Small is beautiful: discovering the minimal set of unexpected patterns." Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2000.

[19] Dang, Xuan Hong, et al. "Local outlier detection with interpretation." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 2013.

[20] Hodge, Victoria, and Jim Austin. "A survey of outlier detection methodologies." Artificial intelligence review 22.2 (2004): 85-126.