Hochschule Darmstadt

- Fachbereich Mathematik und Naturwissenschaften -

Analyse von Algorithmen des maschinellen Lernens für das Mentorensystem Informatik

Abschlussarbeit zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt von
Maxi Neubacher

Abgabedatum: 02.04.2013

Referent : *Prof. Dr. Andreas Thümmel*
Korreferent : Dr. *Frederik Janssen*

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Soweit ich auf fremde Materialien, Texte oder Gedankengänge zurückgegriffen habe, enthalten meine Ausführungen vollständige und eindeutige Verweise auf die Urheber und Quellen. Alle weiteren Inhalte der vorgelegten Arbeit stammen von mir im urheberrechtlichen Sinn, soweit keine Verweise und Zitate erfolgen.

Mir ist bekannt, dass ein Täuschungsversuch vorliegt, wenn die vorstehende Erklärung sich als unrichtig erweist.

Darmstadt, April 2013 Maxi Neubacher

Abstrakt

Das Mentorensystem im Fachbereich Informatik an der TU Darmstadt wurde im Jahr 2012 um eine Maßnahme erweitert. Studierende, die einen geringen Prüfungserfolg im ersten Semester aufweisen, werden im zweiten Semester weiter betreut. Die Studierenden, auf die das zu trifft, werden im weiteren „Härtefälle“ genannt. Die Gründe für den Misserfolg werden dabei in einem Einzelgespräch mit der Fachstudienberatung erläutert um zeitnah Abhilfe zu schaffen. Welcher Studierende als „Härtefall“ gilt wird momentan, von der Studienberatung, anhand spezieller Angaben zur Person und Erfahrungswerten entschieden. Diese Angaben werden durch einen Fragebogen bezüglich des Studienverhaltens ermittelt. Um den Aufwand der Datenanalyse zu verringern, soll zukünftig der Computer, anhand von aufgestellten Regeln, diese Tätigkeit übernehmen.

Ziel dieser Arbeit ist es automatisiert solche geeigneten Regeln zu finden, anhand derer die Studierenden als „Härtfall“ oder „kein Härtefall“ kategorisiert werden.

Zur Lösung der Problemstellung wurde in dieser Arbeit Maschinelles Lernen angewandt. Dabei sucht ein Lernalgorithmus nach Gesetzmäßigkeiten in vorhandenen Daten und gibt diese zum Beispiel als Regeln wieder.

Die gesammelten Daten, bestehend aus den ausgefüllten Fragebögen der Studierenden und der Zuordnung der Studienberatung dazu, wurden in einer arff-Datei verarbeitet welche in die Software WEKA eingebunden werden kann.

Vorrangig wurden in dieser Arbeit Regel-Lern-Algorithmen und Entscheidungsbaumlerner, die WEKA bereit stellt, genutzt und deren Algorithmen dahinter genauer betrachtet.

Es wurde beobachtet in wie fern sich Regeländerungen durch Hinzufügen neuer Daten ergeben. Dabei wurden die aktuellen Regelmengen mit Ergebnissen aus vorangegangenen Arbeiten verglichen. Hierbei hat sich gezeigt, dass es keine allgemeingültigen Regelmengen gibt, da sich durch den neuen Datensatz komplett neue Regelmengen ergeben haben.

Die aktuell gefundenen Modelle wurden nach bestimmten Gütekriterien beurteilt und die einzelnen Regeln interpretiert.

Das Ergebnis dieser Betrachtungen ist, dass das Ziel dieser Arbeit momentan noch nicht realisierbar ist. Die betrachteten Klassifikationen ergaben zwar sinnvolle und

nachvollziehbare Regeln, lieferten aber noch keine ausreichend genaue Vorhersage bezüglich der Problemstellung. Die Klassifizierer, die Lern-Algorithmen bei der Regelfindung verwenden, verbessern sich mit steigender Anzahl an Daten. Um ein finales Modell zu erhalten, welches alle zukünftigen Studierenden ausreichend zutreffend klassifiziert, nach „Härtefall“ oder „kein Härtefall“, benötigen die Klassifizierer mehr, zu diesem Zeitpunkt noch nicht vorhandene, Daten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Das Mentorensystem [1] im Fachbereich Informatik an der TU Darmstadt wurde im Wintersemester 2006/07 eingeführt. Dieses richtet sich zunächst an die Studierenden des ersten Semesters, um ihnen den Einstieg in das studentische Leben zu erleichtern. Jedem Erstsemestler wird dabei ein Mentor zugewiesen, der dem Studierendem als Ansprechpartner bei Problemen zur Seite steht. Vorrangige Themen sind dabei studienalltägliche, organisatorische und prüfungsvorbereitende Probleme. Seit dem Wintersemester 2007/08 entstand daraus ein Kooperationsprojekt mit der Hochschuldidaktischen Arbeitstelle der TU Darmstadt [2]. Das Hauptziel dieses Projektes ist das frühzeitige Erkennen von Problemen, die bei Studierenden auftreten können, um daraufhin schnellstmöglich mit geeigneten Lösungen reagieren zu können. Auf diese Weise soll der Studienerfolg garantiert werden. Seit dem Sommersemester 2010 wurde das Mentorensystem um die Maßnahme erweitert, dass nun auch einige ausgewählte Studierende, über das erste Semester hinaus, weiterhin im zweiten Semester betreut werden. Dabei sollen vor allem diejenigen Studierenden, die im ersten Semester nur einen geringen Prüfungserfolg aufweisen konnten, weiterhin betreut werden. Hierbei handelt es sich um die so genannten „Härtefälle“. Durch ein intensives Beratungsgespräch mit einem Mitarbeiter der Fachstudienberatung sollen die Gründe des Prüfungsmisserfolges erläutert werden, um danach gezielt Abhilfe zu schaffen. Die Kapazität der Studienberatung lässt es nicht zu mit allen Studierenden ein Einzelgespräch zu führen. Deshalb stellt sich die Frage, wer von den Studierenden zu einem Beratungsgespräch eingeladen werden sollte und wer nicht. Bisher entscheidet die Studienberatung noch bei jedem einzelnen Studierenden anhand von speziellen Angaben zu dieser Person, in welche Kategorie derjenige eingestuft wird. Da die Erstsemesterzahlen im Studiengang Informatik jährlich steigen, wird der Aufwand immer größer bei jedem Studierenden eine Einzelentscheidung zu treffen, ob ein „Härtefall“ oder „kein Härtefall“ vorliegt. Um diesen Aufwand der Datenanalyse zu verringern, wird geplant ein Programm zu erstellen, welches automatisiert die Liste der zu beratenden Studierenden erzeugt. Was die Studienberatung mittels Erfahrung

bisher gemacht hat, soll zukünftig der Computer unter Verwendung von Regeln übernehmen. Ziel dieser Arbeit ist es mögliche Regeln dazu zu finden und diese anhand von bestimmten Kriterien zu beurteilen.

1.1 Gliederung

In Kapitel 2 werden zunächst die grundlegenden Begriffe erklärt, die in dieser Arbeit verwendet werden. Dazu gehört das Themengebiet des Maschinellen Lernens, die verwendete Software WEKA sowie die verarbeitete arff-Datei. Zusätzlich werden die Grundlagen einer Klassifikation erläutert.
Kapitel 3 befasst sich zum einen mit der Datengrundlage dieser Arbeit und zum anderen mit der Ergebnispräsentation bestehender Arbeiten zum Thema dieser Arbeit.
In Kapitel 4 wird die Klassifikation in WEKA genauer erklärt. Dabei werden die Gütekriterien vorgestellt sowie die unterschiedlichen Regel-Lern-Algorithmen und Entscheidungsbaumlerner näher betrachtet.
Die Probleme die sich bei der Datenvorbehandlung ergeben haben und die Lösungen derselben werden in Kapitel 5 dargestellt. Weiterhin werden in diesem Kapitel die Ergebnisse der Arbeit in Bezug auf die Problemstellung präsentiert.
In Kapitel 6 befinden sich eine Zusammenfassung, ein Fazit und Ausblick für mögliche weitere Betrachtungen dieser Arbeit.

2. Grundlagen

Das folgende Kapitel bietet eine Übersicht über grundlegende Begriffe, die in dieser Arbeit verwendet werden.

2.1 Maschinelles Lernen

Maschinelles Lernen [3] ist ein Forschungsgebiet der Informatik und befasst sich mit Computerprogrammen, die aus Erfahrung lernen. Ein Programm lernt anhand von gegebenen Daten Gesetzmäßigkeiten darin zu erkennen. Maschinelles Lernen wird in vielen Gebieten angewendet, wie zum Beispiel bei der Handschrifterkennung [4], bei der Steuerung von Robotern [4] sowie bei Diagnosen in der Medizin [4]. In dieser Arbeit nutze ich Methoden des Maschinellen Lernens zur Lösung der Problemstellung.

2.2 WEKA

Das für diese Arbeit verwendete Softwaretool heißt „Waikato Environment for Knowledge Analysis“, kurz WEKA [5], und ist frei verfügbar. WEKA
wurde an der University of Waikato (Neuseeland) entwickelt, ist in Java geschrieben und stellt verschiedene Techniken aus dem Bereich des Maschinellen Lernens bereit.

2.3 arff-Datei

Die verwendeten Daten dieser Arbeit habe ich im Attribut-Relation File Format (arff) [6] verarbeitet und diese anschließend in WEKA eingebunden. Dabei werden beobachtete Beispiele in Instanzen geschrieben. Jeder einzelne Studierende wird

somit durch genau eine Instanz beschrieben. Jede Instanz besteht aus mehreren Attributen, die Eigenschaften einer Instanz. Attribute sind die bestimmten Merkmale zu einem Studierenden. WEKA erkennt vier verschiedene Arten von Attributtypen: beliebige Zeichenfolgen (string), Datumsangaben (date), numerische Attribute (numeric), und nominale Attribute (nominal). Numerische Attribute lassen Zahlenwerte mit natürlicher Ordnung als Merkmalsausprägungen zu, wobei logische Operatoren wie „gleich“, „ungleich“ aber auch „größer als“ und „kleiner als“ angewandt werden können. Nominale Attribute lassen nur die explizit vorgegeben Merkmale zu, die lediglich durch „gleich“ und „ungleich“ verglichen werden können. Nominale Attribute, die nur zwei Werte zulassen, nennt man binär. Die Menge aller Instanzen, die mit einer gemeinsamen Menge von Attributen beschrieben werden bilden einen Datensatz (Relation). Eine arff-Datei kann jeweils nur eine Relation enthalten. Eine arff-Datei besteht aus einem Kopf (header) und dem dazugehörigen Körper (body). Der Kopf der Datei beginnt mit dem Namen derselben, danach folgt die Deklaration der Attribute und endet mit der Einleitung der Instanzen. In der Abbildung 2.1 wird dies verdeutlicht.

```
@relation Relationenname
@attribute Attributname1 Attributtyp
@attribute Attributname2 Attributtyp
…
@data
```

Abbildung 2.1: Deklaration einer arff-Datei

Bei nominalen Attributen werden an Stelle des Attributtyps die möglichen nominalen Werte in geschweifte Klammern gesetzt.
Nach dem Kopf schließt der Körper der Datei an, die eigentlichen Daten. Hier werden alle genutzten Instanzen nacheinander aufgelistet. Jede Instanz füllt mit ihren Attributwerten, die durch Kommas getrennt werden, eine Zeile aus. Die Reihenfolge der Attributwerte muss mit der Reihenfolge in der Deklaration der Attribute übereinstimmen. Leerzeichen werden durch Unterstriche ersetzt und fehlende Werte mit einem Fragezeichen.

Neben dem WEKA eigenem Dateiformat arff kann der Datenimport in WEKA auch über eine CSV-Datei [7] oder einer Datenbankverbindung mittels SQL [8] erfolgen.

2.4 Klassifikation

Die Umsetzung von Maschinellem Lernen kann durch verschiedene Algorithmen erfolgen, wobei man zwischen überwachtem Lernen und unüberwachtem Lernen unterscheidet. Beim unüberwachten Lernen ist der Zielwert der Daten nicht bekannt. Es besteht hier die Möglichkeit einer Clusteranalyse [9], bei der ähnelnde Instanzen in Gruppen eingeteilt werden, oder einer Assoziationsanalyse [10], bei der Abhängigkeiten unter den Attributen ermittelt werden. In dieser Arbeit verwende ich die Methode der Klassifikation, die unter den Begriff des überwachten Lernens fällt. Im Gegensatz zum unüberwachten Lernen gibt es bei der Klassifikation eine Menge an Beispielen deren Zuordnung schon bekannt ist [4].
Das Resultat der Zuordnung ob ein „Härtefall“ oder „kein Härtefall“ vorliegt wissen wir bereits durch die Einteilung der Studienberatung. Bei einer Klassifikation werden durch die erhobenen Daten und die Kenntnis der Zuordnung darüber, Gesetzmäßigkeiten ermittelt. Die Gesetzmäßigkeiten stellen eine Abhängigkeit der Daten mit ihrer Zuordnung dar. Anhand dessen werden zukünftige, nicht zugeordnete Daten, klassifiziert.
Bei der Klassifikation wird zunächst ein ausgewählter Lernalgorithmus mit einem Teil der bereits klassifizierten Instanzen „trainiert“. Während dieser Phase des Lernens wird ein Modell konstruiert, welches den Zusammenhang zwischen den Attributwerten und ihrer Zuordnung beschreibt.
Nachdem das Modell konstruiert wurde, wird es mit einer Testmenge getestet. Dabei werden die Beispiele aus dieser Menge ohne Berücksichtigung ihrer Zuordnungen anhand des Modells klassifiziert. Anschließend werden die so entstandenen Klassifikationen mit den tatsächlichen verglichen um zu sehen wie gut das Modell Vorhersagen zutreffend liefert.
Da man oftmals keine seperate Testmenge zur Verfügung hat besteht in WEKA die Möglichkeit die Instanzmenge in eine Trainings- und eine Testmenge aufteilen zu lassen. Die Aufteilung der Menge kann durch zwei Arten erfolgen. Zum einen kann

man den Anteil der Trainings- und Testmenge prozentual eingeben, zum anderen kann eine Kreuzvalidierung [11] durchgeführt werden. Bei letzterer wird eine Datenmenge in x-gleichgroße Teile zerlegt. Jede Teilmenge besteht dadurch aus derselben Anzahl an Instanzen. (x-1) der Teilmengen werden dabei zusammen als Trainingsmenge genutzt, der restliche Teil als Testmenge. In dieser Arbeit habe ich mich bei den Ausführungen der Klassifikationen für eine, üblicherweise verwendete [12], 10-fache Kreuzvalidierung entschieden. Dabei werden die Instanzen in zehn Teilmengen unterteilt, wobei neun davon als Trainingmenge und ein Teil als Testmenge genutzt wird. Dieses Szenario wird zehnmal wiederholt, so dass jede Teilmenge einmal getestet wird.

3. Datengrundlage

Dieses Kapitel gibt einen Überblick darüber wie die verwendeten Daten zustande kamen. Des Weiteren wird erläutert wie die Daten in einer arff-Datei verarbeitet werden. Weiterhin werden Ergebnisse früherer Arbeiten bezüglich des Themas dieser Arbeit präsentiert.

3.1 Der Fragebogen

Nachdem die Studierenden das erste Fachsemester absolviert, die Prüfungen geschrieben haben und die Klausurergebnisse bekannt sind, sind sie, laut Prüfungsordnung, dazu verpflichtet einen Fragebogen, mittels dem Onlinesystem Moodle [13], auszufüllen. Dieser Fragebogen beinhaltet außer allgemeinen Angaben zur Wohnsituation und der vorher besuchten Schule auch Fragen zu den aktuell absolvierten Prüfungen, dem Studienverhalten, zu subjektiven Gründen des Prüfungsmisserfolgs sowie zur momentanen privaten Situation. Der Fragebogen ist Teil des Mentorensystems und wurde entwickelt, um zeitnah etwas über die möglichen Probleme des Prüfungsmisserfolges der Studierenden zu erfahren. Die Angaben aus den Fragebögen bilden die Grundlage aller weiteren Betrachtungen. Die dadurch entstandenen Daten werden in eine arff-Datei geschrieben, in WEKA eingebunden um letztendlich eine Klassifikation durchführen zu können.
Zu Beginn des Projektes bestand ein Fragebogen aus 52 Fragen, wobei für die Klassifikation, ob ein „Härtefall“ oder „kein Härtefall“ vorliegt, nicht alle Fragen dafür relevant erschienen. Beispielsweise beinhaltet der Fragebogen auch Angaben zu Name und Matrikelnummer der Studierenden. Diese oder ähnliche Angaben sollten nicht bei der Klassifikation mit berücksichtigt werden, da sie keine Auswirkungen auf die Zuordnung haben. Von dem Namen eines Studierenden kann man nicht daraus schließen ob dieser eine Beratung nötig hat. Aus diesem Grund wurde entschieden 45 der Fragen als Merkmal für die Klassifikation in die arff-Datei mit aufzunehmen.

Zwischenzeitlich haben sich Änderungen im Fragbogen ergeben, auf die ich im Kapitel 5.1 weiter eingehen werde. Jede Frage aus dem Fragebogen wird durch genau ein Attribut in der arff-Datei beschrieben. Somit liegen für jeweils einen Studierenden 45 Angaben zur Person vor, anhand derer der Studierende klassifiziert wird. Von den 45 Attributen wurden 34 Attribute als nominal deklariert, bei diesen Fragen wurden im Fragebogen Antwortmöglichkeiten dazu gegeben. 7 Attribute wurden numerisch und 4 Attribute binär beschrieben. In der Tabelle 3.1 sind alle verwendeten Attribute nach der Kategorie im Fragebogen mit Name und Typ aufgelistet. Das in der arff-Datei zuletzt eingefügte Attribut bezeichnet man als die Klasse und entspricht der Information auf die geschlossen werden soll. Da danach gefragt wird, welcher der Studierenden beraten werden sollte, ist in diesem Fall das Attribut *„Beratung“* die Klasse. Vom Wintersemester(WS) 2010/11 und Sommersemester(SS) 2011 haben 123 Studierende einen solchen Fragebogen ausgefüllt und wurden anhand dessen von der Studienberatung eingestuft, ob eine persönliche Beratung sinnvoll ist oder nicht. Von den 123 Studierenden wurden 35 als „Härtefall“ klassifiziert, die restlichen 88 wurden als „kein Härtefall“ bewertet.
Jeder Fragebogen eines Studierenden wird durch eine Instanz, mit den dazugehörigen 45 Attributen, in der arff-Datei beschrieben. Die Fragebögen vom Wintersemester und Sommersemester unterscheiden sich, da die angebotenen Veranstaltungen andere sind. In die arff-Datei wurden alle Vorlesungen mit aufgenommen damit auch alle bei der Klassifikation berücksichtigt werden. Da in den einzelnen Instanzen jeweils ein Wert aller deklarierten Attribute vorhanden sein muss, entstehen fehlende Werte. Diese werden in der Instanz durch ein Fragezeichen kenntlich gemacht. Bei den Studierenden die zu einem WS begonnen haben wurde ein Fragzeichen an der Stelle der Instanz geschrieben, bei der nach den Vorlesungen des SS gefragt wurde und umgekehrt.

Tabelle 3.1: Die verwendeten Attribute zur Klassifikation

Kategorie	**Attributname**	**Attributtyp**
allgemeine Angaben	Anfahrtsweg	nominal
	Abinote	nominal
	Mathenote	nominal
Angaben zur Prüfungsanmeldung	# zugelassene Prüfungen	numerisch
	# angemeldete Prüfungen	numerisch
	# geschriebene Prüfungen	numerisch
	# abgemeldete Prüfungen	numerisch
	# entschuldigte Prüfungen	numerisch
	# bestandene Prüfungen	numerisch
	# Klausureinsichten	numerisch
Angaben zur Prüfungsvorbereitung	Häufigkeit der besuchten Übungen (GdI1,FGdI1,HCS,TGdI,Mathe1)	nominal
	Häufigkeit der gemachten Hausübungen (GdI1,FGdI1,HCS,TGdI,Mathe1)	nominal
	Häufigkeit der besuchten Vorlesungen (GdI1,FGdI1,HCS,TGdI,Mathe1)	nominal
Angaben zum Studienverhalten	Studienaufwand Lehrveranstaltungen	nominal
	Studienaufwand Selbststudium	nominal
	Lerngruppe	binär
	Job	binär
	Stunden/Woche Job	nominal
	Stunden/Woche Hobby	nominal
	Stunden/Woche Freunde/Familie	nominal
	Sunden/Tag Computerspiele	nominal
	Stunden/Tag Onlinenetzwerke	nominal
	Stunden/Tag TV	nominal
Angaben zu Gründen des Prüfungsmisserfolges	Lernstoff zu schwer	nominal
	Lernstoff zu viel	nominal
	Lernstoff anders als erwartet	nominal
	zu wenig Zeit investiert	nominal
	zu viel vorgenommen	nominal
	hatte keinen Lernplan	nominal
	Klausurinhalte anders als erwartet	nominal
	zu nervös in Klausur	nominal
	Lehrveranstaltung hat nicht auf Klausur vorbereitet	nominal
Private Situation	Private Gründe	binär
Klassifikation	Beratung	binär

3.2 Vorherige Ergebnisse

Die vorhandenen Daten vom WS 2010/11 sowie SS 2011 wurden in einer früheren Arbeit zu diesem Thema [14] in eine arff-Datei geschrieben. Während einer weiteren vorangegangenen Arbeit [15] wurde die arff-Datei in WEKA eingebunden um einige Klassifikationen durchzuführen. Dabei wurden mehrere verschiedene Lernalgorithmen verwendet und die jeweiligen gefundenen Regelmengen genauer untersucht.

Zudem wurden während der erstgenannten Arbeit [14] experimentell Versuche unternommen verschiedene Möglichkeiten der Aufteilung von Trainings- und Testmenge zu nutzen und hinsichtlich der Auswertungen zu vergleichen. Dazu gehören verschiedenartige Kreuzvalidierungen sowie unterschiedliche prozentuale Aufteilungen. Aus dem Grund, dass eine 10-fache Kreuzvalidierung die besten Ergebnisse geliefert hat, habe ich mich bei den weiteren Betrachtungen, in dieser Arbeit, auf diese Aufteilung beschränkt.

Die Maßnahme, dass einige ausgewählte Studierende auch im zweiten Semester weiter durch ein Beratungsgespräch betreut werden, ist relativ neu. Die Tatsache, dass, zum Zeitpunkt der vorherigen Arbeiten, erst zwei Semester von dieser Maßnahme betroffen waren, spiegelte sich in der geringen Anzahl an verfügbaren Daten wieder. Es wurden verschiedene sinnvoll erscheinende Modelle zur Klassifikation gefunden. Dennoch ist es fraglich ob diese Modelle ausreichen, um zukünftige Daten daran zu klassifizieren. Die vorhandene Anzahl an Daten der 123 Studierenden aus zwei Semestern ist zu gering um auf allgemeingültige Gesetzmäßigkeiten zu schließen. Diese Befürchtung wurde durch die, an den Modellen, getesteten Instanzen und die Auswertungen dazu bestätigt. Es stellte sich heraus, dass keine der gefundenen Modelle eine ausreichend genaue und zutreffende Vorhersage liefert.

4 Auswertungen in WEKA

In diesem Kapitel wird die Klassifikation in WEKA genauer erklärt. Dabei werden die Gütekriterien vorgestellt sowie die unterschiedlichen Regel-Lern-Algorithmen und Entscheidungsbaumlerner näher betrachtet.

4.1 Gütekriterien

Wie schon zuvor erwähnt, wird bei einer Klassifikation zunächst, anhand einer Trainingsmenge, ein Klassifikationsmodell trainiert. Das so entstandene Modell wird in WEKA ausgegeben. Als Beispiel einer solchen Auswertung, habe ich im Folgenden den Klassifizierer „JRip“ (mit Pruning) mit einer 10-fachen-Kreuzvalidierung genommen. Dieses Modell ist durch die früheren Daten der Studierenden des WS 10/11 und SS 11 entstanden und in Abbildung 4.1 zu sehen.

```
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

JRIP rules:
===========

(prüfungen_bestanden <= 0) and (prüfungen_mitgeschrieben <= 0) =>
beratung=ja (28.0/12.0)
 => beratung=nein (95.0/19.0)

Number of Rules : 2


Time taken to build model: 0.03 seconds
```

Abbildung 4.1: Vorhersagemodell von JRip

Die Gesetzmäßigkeiten, die der Klassifizierer hier in dem Datensatz der Trainingsmenge erkannt hat, wurden als Regeln ausgegeben. Nach jeder Regel wird

in WEKA die Anzahl der Instanzen beschrieben, die durch diese Regel abgedeckt werden (Werte in Klammern). Der erste Wert ist dabei die Gesamtanzahl der Beispiele, die Instanzen, auf die diese Regel zutrifft. 28 der 123 Instanzen erfüllen die Kriterien der Regel und werden somit zur Klasse *beratung=ja* klassifiziert. Der zweite Wert in der Klammer gibt die Anzahl der negativen Beispiele, der Fehlklassifikation davon an. 12 der 28 Instanzen wurden durch die Regel zur Klasse *beratung=ja* eingestuft, da sie die Kriterien erfüllen, gehören allerdings tatsächlich, laut Datensatz, in die andere Klasse. Die Anzahl der abgedeckten positiven Beispiele, die korrekt klassifizierten Instanzen, sollte möglichst hoch sein. Je mehr Instanzen von einer Regel positiv abgedeckt werden, desto besser erscheint diese. Wenn eine Regel mehrere Instanzen korrekt beschreiben kann ist sie allgemeingültiger und die Wahrscheinlichkeit höher, dass auch neue Instanzen dadurch korrekt klassifiziert werden. Die Anzahl der abgedeckten negativen Beispiele dagegen sollte möglichst klein sein.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          77               62.6016 %
Incorrectly Classified Instances        46               37.3984 %
Kappa statistic                         -0.005
Mean absolute error                      0.413
Root mean squared error                  0.5199
Relative absolute error                100.9738 %
Root relative squared error            115.1443 %
Total Number of Instances              123

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.2       0.205       0.28      0.2        0.233      0.48     ja
                 0.795     0.8         0.714     0.795      0.753      0.48     nein
Weighted Avg.    0.626     0.631       0.591     0.626      0.605      0.48

=== Confusion Matrix ===

  a  b   <-- classified as
  7 28 |  a = ja
 18 70 |  b = nein
```

Abbildung 4.2: Auswertung des Vorhersagemodells von JRip

Nachdem ein Klassifikationsmodell konstruiert wird, wird dieses anhand einer Testmenge getestet. Ein beispielhaftes Ergebnis einer solchen Auswertung ist in Abbildung 4.2 dargestellt. WEKA liefert hier mehrere verschiedene Gütekriterien [16], die die Qualität der gefundenen Regelmenge beurteilen. Dazu gehören beispielsweise die Klassifikationsrate oder die Konfusionsmatrix eines gefundenen Modells. Im Folgenden werde ich die einzelnen Gütekriterien genauer erklären.

Die Klassifikationsrate (Correctly Classified Instances) ist der prozentuale Wert der Anzahl korrekt klassifizierter Instanzen, gemessen an der Gesamtmenge der getesteten Instanzen. Die Klassifikationsrate liefert eine Genauigkeitsabschätzung in wie weit ein Modell Vorhersagen zutreffend liefert.
Die Konfusionsmatrix gibt einen weiteren Einblick über den Wahrheitsgehalt der getesteten Instanzen bezüglich ihrer Klassifikationen. Die Darstellung einer solchen Matrix ist in der Tabelle 4.1 zu sehen.

Tabelle 4.1: Die Konfusionsmatrix

a	b	← classified as
TP	FN	a = ja
FP	TN	b = nein

Der *True-Positive-Wert (TP)* ist die Anzahl der getesteten Instanzen, die nach dem Modell in die Kategorie *beratung=ja* eingestuft wurden und auch tatsächlich zu dieser Gruppe gehören. Der *True-Negative-Wert (TN)* ist die Anzahl der Instanzen, die korrekterweise zur Klasse *beratung=nein* kategorisiert wurden. Die beiden Werte sollten möglichst groß sein. *False-Positive (FP)* und *False-Negative (FN)* sind Werte über die Anzahl der Instanzen die fälschlicherweise einer Klasse zugeordnet worden sind. Der *FP-Wert* gibt die Anzahl der Instanzen wieder, die der Klasse *beratung=ja* zugeordnet wurden, aber tatsächlich der Klasse *beratung=nein* angehören. Die Anzahl der Instanzen die zur Klasse *beratung=nein* eingeordnet wurden, aber tatsächlich zur Klasse *beratung=ja* gehören werden durch den *FN-Wert* beschrieben. Dieser Wert sollte besonders klein ausfallen, da es schwerwiegend wäre einen Studierenden als „kein Härtefall“ zu deklarieren obwohl er einer ist.

Die Raten der eben genannten Werte werden auch explizit in WEKA angegeben. Sie werden durch die einzelnen Werte der Matrix an der tatsächlichen Instanzanzahl der jeweiligen Klasse gemessen.
Weitere Gütekriterien, die anhand der Konfusionsmatrix berechnet werden, sind: Precision, Recall, F-Measure, ROC-Area und kappa statistic, auf die ich, in dieser Arbeit, nicht weiter eingehen werde. Zusätzlich gibt WEKA, bei jeder Klassifikation, noch einige statistische Fehlermaße aus.

4.2 Entscheidungsbäume

Eine Klassifikation kann, unter anderem, mittels eines Entscheidungsbaumes [17] durchgeführt werden. Ein Entscheidungsbaum besteht immer aus Knoten, Kanten und Blättern. Im Falle der Klassifikation entspricht jeder Knoten einem Attributwerttest. Dabei wird eins der, in der arff-Datei, vorhandenen Attribute ausgewählt, anhand dessen klassifziert wird. Den ersten Attributwerttest nennt man Wurzelknoten. Jeder Test hat entsprechende Antwortmöglichkeiten, die zugelassenen Werte dieses Attributes, die durch verschiedene Kanten repräsentiert werden und den Pfad zum nächsten Attributwerttest darstellen. Bei nominalen Attributen wird jeder mögliche Antwortwert desselben durch eine Kante dargestellt. Bei numerischen Attributen findet ein Vergleich mit einem Zahlenwert statt. Alle Pfade enden mit einem Blatt, der Klassifikation.
Eine zu klassifizierende Instanz durchläuft den Entscheidungsbaum von oben, bei dem Wurzelknoten beginnend, nach unten. An jedem Knoten findet ein Attributwerttest statt. Der Attributwert der zu klassifizierenden Instanz entscheidet über den weiteren Pfad. Die Instanz folgt der Kante mit dem Ergebnis des Tests und gelangt dadurch zu dem nächsten Attributwerttest. Die Prozedur wird so lange durchgeführt bis die Instanz auf ein Blatt trifft. Der Inhalt des Blattes beschreibt die Klassifizierung dieser Instanz.
Die Abbildung 4.3 zeigt eine schemenhafte Darstellung eines Entscheidungsbaumes mit einer möglichen Klassifikation.
Bei der Konstruktion eines Entscheidungsbaumes wird die so genannte Divide- and Conquer (Teile und Herrsche) Strategie [18] benutzt. Bei dieser Strategie wird ein

Problem in kleinere Teilprobleme aufgeteilt (Divide-Schritt). Die Teilprobleme werden rekursiv in weitere Probleme aufgeteilt bis eine Lösung gefunden wird (Conquer-Schritt). Am Ende werden die Teillösungen zu einer Gesamtlösung zusammengefügt. Dabei wird zunächst ein Attribut als Wurzelknotentest gewählt. Die möglichen Attributwerte dessen ergeben jeweils eine Verzweigung. Die Trainingsmenge wird anhand ihrer unterschiedlichen Attributwerte in Teilmengen aufgeteilt. Jede Teilmenge wird erneut, nach Auswahl eines geeigneten Attributs, daran getestet und wiederum aufgeteilt. Diese Verzweigungen werden so lange durchgeführt bis in jeder Teilmenge nur noch Instanzen mit derselben Klasse enthalten sind. Diese Klasse wird am Ende jedes Pfades als Blatt symbolisiert.
Die Attributauswahl während der Trainingsphase wird von den einzelnen Klassifizierern unterschiedlich gehandhabt. Die Algorithmen benutzten dabei verschiedene Maße zur Bestimmung geeigneter Attribute.

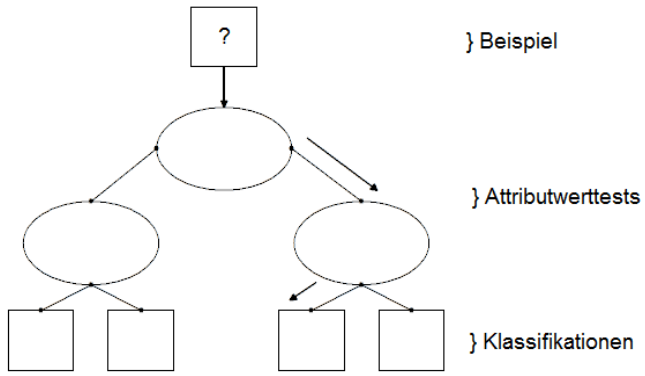

Abbildung 4.3: Klassifikation durch Entscheidungsbaum, eigene Darstellung

4.3 Entscheidungsbäume in WEKA

WEKA bietet Klassifizierer an, die Entscheidungsbäume zur Klassifikation verwenden. In dieser Arbeit habe ich den Klassifizierer „J48“ [19] verwendet und werde im Folgenden genauer auf dessen Algorithmus eingehen.
„J48“ verwendet, bei der Konstruktion eines Entscheidungsbaumes, den so genannten „C 4.5-Algorithmus“ [19]. Dieser Algorithmus wird genutzt um geeignete Attribute auszuwählen, die im Entscheidungsbaum zur Klassifikation beitragen. Es

wird das Attribut gesucht, dass die meiste Information in Bezug auf die Klasse liefert. Dabei wird als Maß der normierte Informationsgehalt [17] jedes Attributs berechnet. Die so errechneten Werte dienen dann, im Vergleich, bei der Auswahl des „besten“ Attributs. Um diesen Wert zu erhalten, wird zunächst die Entropie [17], der Informationsgewinn, des Datensatzes gemessen. Die Entropie ist die Anzahl von Bits, die benötigt wird um die Klassifikation eines zufällig gezogenen Beispiels aus der Datenmenge zu kodieren.

S= Menge von Trainingsbeispielen, S+= Menge aller positiven Beispiele in S
S-= Menge aller negativen Beispiele in S

$$\text{Entropie}(S) = -\frac{|S+|}{|S|}\log_2\frac{|S+|}{|S|} - \frac{|S-|}{|S|}\log_2\frac{|S-|}{|S|}$$

Die Entropie lässt sich, wie oben, für die gesamte Trainingsmenge berechnen und auch für die einzelnen Teilmengen. Anschließend wird die erwartete Verringerung der Entropie, nach der Teilung durch das Attribut A, berechnet. Der Informationsgewinn des Attributes A, Gain(S,A), ist die Differenz der ursprünglichen Information vor dem Test von A und der Restinformation nach dem Test von A.

Sv= Teilmenge von S, für die das Attribut A den Wert v hat

$$\text{Gain}(S,A) = \text{Entropie}(S) - \sum_{v \in Werte(A)} \frac{|Sv|}{|S|}\ \text{Entropie}(Sv)$$

Der Informationsgewinn wird für jedes Attribut berechnet. Die Sortierung der Attribute mit Hilfe dieser Information favorisiert Attribute mit vielen verschiedenen Werten. Bei einem Attribut mit sehr vielen möglichen Antwortwerten kann es beispielsweise im Extremfall dazu kommen, dass das Attribut für jedes Beispiel einen eigenen Wert besitzt und dadurch sehr viele Teilmengen entstehen. Dieses Attribut hätte dann den höchsten Informationsgewinn. Allerdings wäre es für die Vorhersage nicht geeignet, da in diesem Fall keine Verallgemeinerung der Trainingsmenge vollzogen wird, sondern nur auswendig gelernt wird. Deshalb wird der Informationsgewinn normalisiert mit:

SplitInformation(S,A) = - $\sum_{v \in Werte(A)} \frac{|Sv|}{|S|} \log_2 \frac{|Sv|}{|S|}$

Dadurch ergibt sich das eigentliche Maß:

GainRatio(S,A) = $\frac{Gain(S,A)}{SplitInformation(S,A)}$

Das Attribut mit dem größten GainRatio-Wert wird als das „beste Attribut“ ausgewählt mit in den Entscheidungsbaum aufgenommen.

4.4 Regellerner

Eine Regel beschreibt den Zusammenhang der Attribute mit der Klassenzuordnung in einem Datensatz und besteht aus einem Körper und dem dazugehörigen Kopf. Der Körper ist dabei die Menge an Bedingungen, den Attributwerttests, welche erfüllt werden müssen damit eine Instanz in die vorhergesagte Klasse eingeordnet wird. Bei einem Attributtest wird gefordert, dass ein Attribut einen bestimmten Wert aus dessen Wertebereich annimmt. Der Kopf repräsentiert die Zuordnung bei Erfüllung der Regel. Die im vorher gegangenen Kapitel erwähnte Klassifizierung ergab folgende Regel:

(prüfungen_bestanden <= 0) and (prüfungen_mitgeschrieben <= 0) => beratung=ja (28.0/12.0)

Eine Instanz wird demnach auf die Attribute *prüfungen_bestanden* und *prüfungen_mitgeschrieben* getestet. Wenn beide Attributwerte *<= 0* sind, wird die Instanz der Klasse *beratung=ja* zugeordnet.

Es gibt aber auch leere Regeln, die keine Bedingungen enthalten, sondern lediglich aus dem Kopf bestehen. Erreicht eine Instanz eine leere Regel, wird sie ohne Prüfung eines Attributwertes sofort in die vorhergesagt Klasse eingeordnet.

Regeln sind durch die einfachen Wenn-Dann-Bedingungen im Wesentlichen sehr gut verständlich und für den Betrachter leicht nachzuvollziehen.

Mehrere Regeln bilden eine Regelmenge. Eine zu klassifizierende Instanz durchläuft dabei die Liste, der Regeln nach, von oben nach unten. Sobald eine Regel auf die Instanz zutrifft, wird diese danach klassifiziert und aus der Instanzmenge genommen.

Trifft keine Regel zu, tritt die zuletzt aufgeführte Standard-Regel in Kraft, die alle bisher nicht abgedeckten Instanzen einer Klasse zuordnet.
Die Algorithmen der Klassifizierer in WEKA unterscheiden sich in der Art wie sie Regeln lernen. Die regelbasierten Klassifizierer benutzen dabei die Seperate- and Conquer Strategie [20]. Es wird eine Regel gelernt und alle Beispiele, die von dieser Regel abgedeckt werden, werden vom Datensatz entfernt (Seperate-Schritt). Danach werden neue Regeln gelernt, wobei die beste, unter bestimmten Kriterien, zu dem Regelsatz hinzugefügt wird (Conquer-Schritt). Die beiden Schritte werden iterativ so lange wiederholt bis keine positiven Beispiele mehr im Datensatz vorhanden sind.

4.5 Overfitting/ Pruning

4.5.1 Overfitting

Während einer Klassifikation lernt ein Algorithmus anhand der Trainingsdaten Regeln, die diese Daten beschreiben. Die Regeln sollten in komprimierter, verallgemeinerter Form den Datensatz wieder geben. Dabei kann es passieren, dass der Algorithmus keine allgemeinen Regeln findet, die gleichzeitig mehrere Beispiele beschreibt. Er lernt demnach die Trainingsdaten auswendig und gibt diese genau so wieder, anstatt wie gewünscht zu verallgemeinern. Dieses Phänomen nennt man Overfitting [21]. Der extremste Fall einer solchen Überanpassung der Daten ist, wenn jedes Beispiel von genau einer Regel abgedeckt wird. Ein solches Modell ist dann sehr genau in der Beschreibung, da die Trainingdaten unverändert wiedergegeben werden. Das Problem bei Overfitting ist, dass noch nicht klassifizierte Beispiele anhand dieser speziellen Regeln klassifiziert werden. Die noch nicht klassifizierten Instanzen müssten demnach genau so in der Trainingsmenge schon vorhanden sein, um korrekt klassifiziert zu werden. Andernfalls werden neue Instanzen, die nicht in der Trainingsmenge beschrieben werden, von keiner Regel abgedeckt und fallen somit automatisch in die Standard-Regel.

4.5.2 Pruning

Um Overfitting zu vermeiden und somit allgemeingültigere Regelmengen oder Entscheidungsbäume zu erhalten gibt es bei einigen Klassifizierern in WEKA die Möglichkeit des Prunens [22]. Durch bestimmte Abbruchbedingungen wird beim Prunen eine Regelmenge oder einzelne Regeln beschnitten. Dadurch werden einige Bedingungen gelockert, was zur Folge hat, dass mehr Beispiele durch eine Regel abgedeckt werden.

Während bei überangepassten Regeln, die durch Overfitting entstehen, die Genauigkeit sehr hoch ist, kommt es bei geprunten Regeln auch zu Fehlklassifikationen. Allerdings können durch die verallgemeinerten Regeln beim Pruning neue, nicht klassifizierte Instanzen besser eingeordnet werden.

Auch die Art, wie die einzelnen Klassifizierer beim Prunen vorgehen unterscheidet sich. Es gibt die Möglichkeit des Pre-Prunings, Post-Prunings oder eine Mischung aus beiden genannten.

Beim Pre-Pruning wird während der Lernphase beschnitten. Eine Regel wird gelernt und beschnitten bis ein bestimmtes Stopkriterium darüber entscheidet wann diese hinreichend genug verfeinert ist.

Beim Post-Pruning wird zunächst eine komplette Regelmenge gelernt und anhand eines bestimmten Qualitätsmaßes bewertet. Anschließend wird die gefundene Hypothese so lange vereinfacht, bis die Bewertung der Vereinfachung schlechter ist als das Maß der ungeprunten anfänglichen Regelmenge. Die Vereinfachung geschieht beim Post-Pruning durch Verwenden geeigneter Pruning-Operatoren.

Pre- und Postpruning lassen sich aber auch miteinander kombinieren, was in einigen Algorithmen genutzt wird.

4.6 Regellerner in WEKA

WEKA stellt verschiedene regelbasierte Klassifizierer zur Verfügung. In diesem Kapitel werde ich genauer auf die einzelnen Algorithmen der genutzten Klassifizierer eingehen.

„ZeroR“ [23] ist zwar, in WEKA, unter den regelbasierten Klassifizierern eingeordnet, ist allerdings im eigentlichen Sinne kein Regellerner. Er lernt keine Regeln sondern gibt lediglich die Mehrheitsklasse der Daten wieder und kann dadurch als Baseline-Klassifizierer verwendet werden. Die Vorhersagegenauigkeit der Auswertung kann als Vergleichswert zu anderen Klassifizierern betrachtet werden.

Der Klassifizierer „OneR“ [24] arbeitet mit einem simplen Algorithmus und gibt genau eine Regel für einen Datensatz aus. Zunächst wird für jedes Attribut der Daten eine Regel aufgestellt. Dabei wird für jeden Attributwert eine Klassifizierung bestimmt. Die Instanzen, die diesen Attributwert besitzen, werden betrachtet und die Mehrheitsklasse, also die Klasse die am öftesten auftritt, als Klassifizierung des Attributwertes ausgegeben. Die Regel, die während der Trainingsphase, die kleinste Fehlerrate aufweist, wird als die „beste“ Regel ausgegeben. Wenn mehrere Regeln dieselbe Fehlerrate besitzen entscheidet der Zufall welche präsentiert wird.

„JRip“ arbeitet mit dem Algorithmus „Repeated Incremental Pruning to Produce Error Reduction (RIPPER)” [25]. Dieser ist eine optimierte Version des “Incremental Reduced Error Pruning (IREP)“-Algorithmus [25]. Aufgrund dessen werde ich zunächst IREP erläutern, bevor ich auf die Erweiterungen zu RIPPER eingehe.

IREP lernt nacheinander Regeln aus der Trainingsmenge. Ist eine Regel gefunden, werden alle Beispiele, die damit abgedeckt werden, aus der Trainingsmenge entfernt. Die nächste Regel wird aus den restlichen Beispielen gelernt. Dies wiederholt sich so lange, bis keine positiven Beispiele mehr in der Trainingsmenge vorhanden sind oder bis eine gefundene Regel eine zu hohe Fehlerrate (> 50%) aufweist.

Regeln werden gebildet indem zunächst die Trainingsmenge in eine Growingmenge (2/3 der Trainingsmenge) und eine Pruningmenge (1/3 der Trainingsmenge) aufgeteilt wird. Auf der Growingmenge wird eine Regel gebildet, indem mit einer leeren Regel begonnen wird und sukzessiv Konditionen dazugefügt werden.

Die Attribute mit maximalem FOIL-InformationGain [26] werden dabei verwendet. FOIL-InformationGain ist der in Kapitel 4.3 beschriebene Informationsgewinn, wobei

dieser noch mit der Anzahl der von dem jeweiligen Attribut abgedeckten positiven Beispiele gewichtet wird.

Die Regel „wächst“ so lange, bis sie keine negativen Beispiele der Growingmenge mehr abdeckt. Nachdem eine solche Regel gefunden ist, wird sie sofort auf der Pruningmenge geprunt. Dabei wird nacheinander eine endliche Anzahl an Konditionen aus der Regel gelöscht. Beginnend mit der Löschung von jeweils einem Attribut bis hin zur Löschung sämtlicher Attribute aus der Regel. Jede Möglichkeit wird anhand der Heuristik $\frac{p+(N-n)}{P+N}$ (P/N= Anzahl positiver/negativer Beispiele in der Pruningmenge; p/n= Anzahl der positiven/negativen Beispiele, die von dieser Regel abgedeckt werden) bewertet. Die mit der maximalen Heuristik wird ausgegeben.

Experimente haben ergeben, dass „IREP“ in der Konstruktion eines solchen Modells sehr schnell ist gegenüber anderen Algorithmen, wie zum Beispiel „C 4.5“ [25]. Die Fehlerrate der gefundenen Modelle dagegen ist relativ hoch [25]. Um letzteres zu optimieren, wurde „IREP“ durch verschiedene Modifikationen ergänzt. Daraus entstand der Algorithmus „RIPPER“.

Im Gegensatz zu „IREP“ wird bei „RIPPER“ die Heuristik $\frac{p-n}{p+n}$, zur Bewertung einer Regel während der Pruningphase, genutzt.

Die Abbruchkriterien, um auszumachen wann genügend Regeln im Regelsatz vorhanden sind, haben sich mit „RIPPER“ erweitert. Zusätzlich zu den bisherigen Kriterien von IREP, der Abbruch wenn keine positiven Beispiele mehr in der Trainingsmenge vorhanden sind oder bis eine gefundene Regel eine zu hohe Fehlerrate (> 50%) aufweist, wurde noch folgendes Stopkriterium hinzugefügt:

- ➔ Description Length (DL) der Regelmenge + verbleibende beispiele >= bisherige DL +64 bits

Decription Length ist die Anzahl in bits, die der Computer benötigt um Daten zu kodieren. Nachdem eine Regel gelernt wurde, wird die Description Length (DL) der Regelmenge und der verbleibenden Beispiele errechnet. RIPPER stoppt das Lernen von Regeln, wenn die so erhaltene DL größer als die kleinste, bis jetzt errechnete DL + 64 Bits ist.

Nachdem eine Hypothese gelernt wurde, findet bei „RIPPER“ noch zusätzlich eine Optimierungsphase statt. Dabei wird für jede gefundene Regel R_i der anfänglichen Hypothese noch zwei alternative Regeln gelernt. Es wird eine neue, andere Growingmenge aus der Trainingsmenge gebildet. Anhand der neuen Menge wird

eine weitere Regel, genannt Replacement R_i^{rep} gelernt und geprunt. Außerdem wird R_i auf der neuen Menge weiter verfeinert, indem zusätzliche Attributtest der Regel angefügt werden. Die so entstandene Regel wird Revision R_i^{rev} genannt. Im Anschluss wird jeweils eine der drei Regeln (R_i, R_i^{rep}, R_i^{rev}) mit in die Regelmenge aufgenommen. Die Regel mit der die gesamte Regelmenge die kleinste DL besitzt, wird als finale Regel der Hypothese hinzugefügt.

Der Klassifizierer „PART“ [27] arbeitet mit einer Kombination aus der Erzeugung von Regeln mit Hilfe eines Entscheidungsbaumes und der Seperate- and Conquer Strategie. Der Algorithmus produziert, mit Hilfe der Seperate- and Conquer Strategie, eine geordnete Regelmenge. Dazu wird zunächst ein geprunter Entscheidungsbaum, aus den Trainingsdaten, konstruiert. Das Blatt mit der größten Abdeckung der Beispiele wird als Regel ausgegeben. Alle dadurch abgedeckten Beispiele werden aus der Menge entfernt. Mit den restlichen Daten der Trainingsmenge wird ein neuer Baum gelernt und wiederum die beste Regel davon ausgegeben. Die Idee dahinter ist, dass jeweils nur das lokale Optimum gesucht wird und nicht, wie bei den anderen Klassifizierern, das globale. Dadurch werden vorschnelle Verallgemeinerungen vermieden.

5 Evaluation

Die Probleme, die sich bei der Datenvorbehandlung ergeben haben und die Lösungen derselben werden in diesem Kapitel dargestellt. Weiterhin werden die Ergebnisse der Arbeit in Bezug auf die Problemstellung präsentiert.

5.1 Datenvorbehandlung

Die in Kapitel 3.2 geschilderten Ergebnisse sind, wie schon erwähnt, Auswertungen der Fragebögen vom Wintersemester 2010/11 und Sommersemester 2011. Zwischenzeitlich, bis zum Beginn dieser Arbeit, haben sich neue Daten angesammelt. Es handelt sich hierbei um die ausgefüllten Fragebögen der Studierenden des WS 11/12 sowie des SS 12. Wie auch schon zuvor wurden die Studierenden anhand der Angaben im Fragebogen von der Studienberatung eingeteilt, ob ein „Härtefall“ oder „kein Härtefall“ vorliegt. Diese Klassifikation wurde ebenfalls noch durch Einzelentscheidungen mit Hilfe von Erfahrungswerten der Studienberatung getroffen.
Die erste Aufgabe meiner Arbeit bestand also darin, die neu dazu gekommenen Daten zu den alten Daten in die arff-Datei mit einzupflegen. Vom WS 11/12 sind 114 ausgefüllte und klassifizierte Instanzen dazugekommen und vom SS 12 noch 29 weitere. Die alte Datei *fragebogen.arff* wurde um 143 Instanzen erweitert, was eine Gesamtinstanzmenge von 266 ergibt. Von den 266 Studierenden wurden 71 als „Härtefall“ klassifiziert, eine persönliche Beratung dieser Studierenden hat sich als sinnvoll und effektiv herausgestellt. 195 der Studierenden dagegen wurden von der Studienberatung so eingestuft, dass eine Einzelberatung nicht zwingend nötig sei.
Da das Projekt „Mentorensystem im FB Informatik“ noch relativ neu ist, wird stets versucht, dieses noch weiter zu optimieren. Eine Maßnahme zur Verbesserung der Vergleichbarkeit der Angaben aus dem Fragebogen des ersten Semesters mit dem zweiten Semester, ergab einige Änderungen im neuen Fragebogen, den die

Studierenden aus dem WS 11/12 und SS 12 erhielten. Die Problematik bestand in der Zusammenführung der Daten von dem alten und neuen Fragebogen.

5.1.1 Änderungen im Fragebogen

Eine Änderung des Fragebogens ist die zusätzliche Frage nach der letzten Schulnote im Fach Informatik. Diese Informationen sind von den Studierenden aus dem WS 10/11 und SS 11 nicht bekannt und circa die Hälfte der Studierenden aus den neuen Semestern hat zu dieser Frage keine Angabe gegeben. Aufgrund dessen habe ich mich entschieden das Attribut „Informatiknote“ erstmal nicht in die arff-Datei mit aufzunehmen. Vorerst würden die wenigen Daten dazu keine aussagekräftige und gefestigte Regel mit sich tragen. Zu einem späteren Zeitpunkt, wenn neue Daten vorhanden sind und ausreichend viele Angaben zu dieser Frage vorliegen, sollte diese dann aber auch mit in die arff-Datei aufgenommen werden.

Des Weiteren hat sich die Skala zur Beantwortung der Frage nach der Abiturnote geändert. Der Teilwertebereich {1.0, 2.0} der Note wurde vorher in drei Teile zerlegt: {1.0-1.3, 1.4-1.7, 1.8-2.0}. Im neuen Fragebogen hat sich der Teilwertebereich der Antwortmöglichkeit zu {1.0-1.5, 1.6-2.0} geändert. Es ist weder möglich die Angaben des früheren Fragebogens, die in diesen Bereich fallen, in die neue Skala einzuordnen, noch umgekehrt. Ich habe mich entschieden, die alte Skala zunächst beizubehalten. Bei den neu hinzugekommenen Studierenden, die {1.0-1.5} oder {1.6-2.0} angekreuzt haben, habe ich an dieser Stelle ein Fragezeichen gesetzt, diese Werte fehlen dann. Das waren aber eher Ausnahmefälle. Falls die neue Skala weiterhin so bestehen bleibt, sollte man sich bei der Einpflegung der kommenden Daten überlegen, diese zu übernehmen und umgekehrt bei den früheren Studierenden, die den Bereich {1.0-2.0} angekreuzt haben, ein Fragezeichen setzen.

Auch der Wertebereich zur Frage, wie viel Stunden pro Woche der Studierende einem Nebenjob nachgeht, hat sich mit der optimierten Version des Fragebogens, verändert. Die Skala hat sich von {weniger_als_5_h, 5-10_h, 11-15_h, 16-20_h, 21-25_h, 25-30_h} zu {weniger_als_5_h, 5-10_h, 11-20_h, 21-30_h, 31-40_h, mehr_als_40_h} geändert. Die Werte aus der alten Skala kann man problemlos in die

neue Skala überführen, da hier lediglich Teilwertebereiche zusammengefasst werden. Aus diesem Grund habe ich den neuen Wertebereich in die neue arff-Datei übernommen. Bei den anfänglichen Instanzen, mit den Attributwerten {11-15_h} und {16-20_h}, habe ich jeweils den Wert auf {11-20_h} gesetzt. Die Instanzen, die die Werte {21-25_h} und {25-30_h} enthielten, habe ich, an Stelle dessen, auf den Wert {21-30_h} gesetzt. Eine Konvertierung der Werte aus der neuen Skala in die alte wäre nicht möglich gewesen. Wenn zum Beispiel ein Studierender im neuen Fragebogen angegeben hat, dass er 11-20 Stunden/Woche arbeitet, könnte das auf der alten Skala entweder 11-15 Stunden oder 16-20 Stunden bedeuten. Eine eindeutige Zuordnung ist also nicht möglich.

Tabelle 5.1 Änderungen im Fragebogen

Fragebogen WS 10/11 + SS 11	**Fragebogen WS 11/12 + SS 12**	**Zusammenführung der Daten in arff-Datei**
-	Informatiknote	-
Skala Abiturnote: {(1.0-1.3),(1.4-1.7),(1.8-2.0),(2.1-2.5),(2.6-3.0),(3.1-3.5),(3.6-4.0),(ab 4.0),(k.A.)}	Skala Abiturnote: {(1.0-1.5),(1.6-2.0),(2.1-2.5),(2.6-3.0),(3.1-3.5),(3.6-4.0),(ab 4.0),(k.A.)}	alte Skala
Skala Nebenjob: {(weniger als 5 h),(5-10 h),(11-15 h),(16-20),(21-25 h),(25-30 h)}	Skala Nebenjob: {(weniger als 5 h),(5-10 h),(11-20 h),(21-30 h),(31-40 h),(mehr als 40 h)}	neue Skala
-	FGdI2_Vorlesung_besucht FGdI2_Übung_gemacht FGdI2_Hausübung_gemacht	drei unterschiedliche Versionen: 1) mit FGdI2, ? bei fehlenden Werten 2) mit FGdI2, Werte von FGdI1 verdoppelt 3) ohne FGdI2

Die aktuell letzte Änderung des Fragebogens ergab sich durch eine Änderung der Prüfungsordnung im Studiengang Informatik. Zuvor wurde die Veranstaltung „Formale Grundlagen der Informatik“ (FGdI) so unterteilt, dass FGdI1 im ersten Semester und FGdI2 im zweiten Semester stattfand. Da sich der, als Grundlage dienende Fragebogen an die Absolventen des ersten Semesters richtet, wurde bei dem alten Fragebogen nur nach der Prüfungsvorbereitung von FGdI1 gefragt. Die

neue Prüfungsordnung besagt nun, dass die Veranstaltungen FGdI1 und FGdI2 beide im ersten Semester stattfinden. Diese Neuigkeit wurde in der aktuellen Verfassung des Fragebogens umgesetzt. Es wird nun zusätzlich, zur Frage nach der Prüfungsvorbereitung von FGdI1 auch nach der Veranstaltung FGdI2 gefragt. Da die Veranstaltung FGdI nur jeweils zum Sommersemester angeboten wird, sind nur die Daten der Studierenden vom SS 11 und SS 12, von dieser Änderung betroffen. Während die Studierenden vom SS 12 im aktuellen Fragebogen Angaben zu FGdI2 gemacht haben, fehlen diese Werte bei den Instanzen des SS 11. Um die bestmögliche Umsetzung für diese Problematik zu finden, habe ich drei unterschiedliche Varianten benutzt und in den Auswertungen verglichen. Zum einen habe ich FGdI2 mit in die arff-Datei aufgenommen und fehlende Werte mit Fragezeichen gekennzeichnet. Bei einer weiteren Variante habe ich statt den Fragezeichen die Werte von FGdI1 verdoppelt. Zum anderen habe ich eine Version ohne FGdI2 geschrieben.

Insgesamt werden im aktuellen Fragebogen drei Fragen zur Prüfungsvorbereitung der Veranstaltung FGdI2 gestellt: nach der Häufigkeit der Übungs- und Vorlesungsbesuche sowie der Häufigkeit der angefertigten Hausübungen. Bei den Varianten, die FGdI2 berücksichtigen ergeben sich also drei weitere nominale Attribute, die in die arff-Datei mit aufgenommen werden müssen: „FGdI2_Vorlesung_besucht“, „FGdI2_Übung_besucht“ und „FGdI2_Hausübung_gemacht“.

Die unbekannten Werte zu diesen Attributen, vom SS 11, habe ich in einer ersten arff-Datei mit einem Fragezeichen gleichgesetzt (version4a.arff). Dadurch wird die Veranstaltung FGdI2 bei einer geeigneten Regelfindung mit berücksichtigt und es werden dabei keine Daten verfälscht. Allerdings sind momentan nur bei 29 der 266 Instanzen die Werte dazu bekannt.

Bei der nächsten Version habe ich versucht genau dieses eben genannte Problem zu umgehen. Ich habe ebenfalls FGdI2 mit in die arff-Datei aufgenommen, aber an Stelle fehlender Werte zu dieser Veranstaltung ein Fragezeichen zu setzen, habe ich jeweils die Angaben zu FGdI1 verdoppelt (version4.arff). Dadurch entstehen Werte zu FGdI2, die die Studierenden aber tatsächlich nicht angegeben haben. Es zeigt lediglich eine hypothetische Wertevergabe auf. FGdI2 wird zwar dadurch bei der Klassifizierung mit berücksichtigt, allerdings sind die Werte zu FGdI2 nicht

wahrheitsgemäß. Diese Version zeigt in etwa, wie eine mögliche Regelmenge aussehen könnte, wenn genügend Werte von FGdI2 vorhanden wären.
Zum anderen habe ich noch eine arff-Datei geschrieben, bei der ich vorerst die Attribute zu FGdI2 weggelassen habe (Version4b). Die Veranstaltung FGdI2 wird hierbei zur Klassifizierung der Studierenden außer Acht gelassen.
Ich habe mit allen drei Varianten Klassifikationen durchgeführt, um zu sehen ob sich Änderungen bezüglich der Regelmengen durch die unterschiedliche Handhabung eines Attributes in der arff-Datei ergeben. Die Frage war, in wie weit dieses Attribut zur Klassifikation durch unterschiedliche Klassifizierer beiträgt.

5.2 Ergebnisse

Wie schon im Kapitel 4 erwähnt, stellt WEKA eine große Anzahl verschiedener Klassifizierer bereit. Aufgrund der Vielzahl an Möglichkeiten zur Klassifikation, musste ich mich bei der Auswertung in dieser Arbeit auf einige wenige davon beschränken. Mein Fokus richtet sich hierbei vor allem auf die, in Kapitel 4.6 beschriebenen, regelbasierten Klassifizierer „ZeroR“, „OneR“, „JRip“ und „Part“. Von den Entscheidungsbäumen habe ich den, in Kapitel 4.3 erläuterten, Klassifizierer „J48“ ausgewertet. In der Rubrik „functions“ in WEKA sind weitere Klassifikationsverfahren zu finden. Um zu sehen ob andere Klassifikationsverfahren eventuell eine bessere Performance liefern als die regel- und Entscheidungsbaumbasierten Klassifzierer, habe ich auch diese in Betracht gezogen. Hier habe ich mich zum einen für den Klassifizierer „SMO“ entschieden, der mit Hilfe von einer Stützvektormaschine klassifiziert. Zum anderen habe ich den Klassifizierer „Multilayer Perceptron“ genutzt, dessen Algorithmus auf der Erzeugung eines künstlichen Neuronalen Netzes beruht. Auf beide zuletzt genannten Verfahren werde ich später noch genauer eingehen.
Von den in Kapitel 4.1 erwähnten, Gütekriterien der Klassifikation in WEKA, gehe ich hier insbesondere auf die Vorhersagegenauigkeit der Klassifzierer ein. Die Vorhersagegenauigkeit ist ein Maß dafür, wie gut ein Klassifizierer bei noch nicht klassifizierten Instanzen funktioniert. Auf die Problemstellung bezogen, zeigt es wie hoch die Wahrscheinlichkeit ist, dass anhand des jeweiligen Klassifizierers die

Studierenden in die richtige Kategorie, nach „Härtefall“ oder „kein Härtefall“, eingeordnet werden. Dieses Maß ermöglicht auch einen Vergleich der Klassifizierer untereinander. Die Vorhersagegenauigkeit sollte möglichst hoch sein. Des Weiteren lege ich besonderes Augenmerk auf die *false-negative-Rate (fn-Rate).* Der *false-negative*-Wert gibt an, wie viele der Instanzen fälschlicherweise durch die gefundenen Regeln der Klasse *beratung=nein* zugeordnet wurden, obwohl sie eigentlich zur Klasse *beratung=ja* gehören. Die *fn-Rate* ist der *false-negative*-Wert gemessen an der Anzahl der Instanzen, die tatsächlich der Klasse beratung=ja angehören. Auf die Problemstellung bezogen, ist die *fn-Rate* besonders wichtig und sollte dementsprechend möglichst klein sein. Es sollte nicht passieren, dass Studierende, bei denen eine Notwendigkeit der Beratung besteht, also zur Klasse beratung=ja gehören, falsch eingestuft werden und letztendlich nicht beraten werden. Gegenteilig betrachtet wäre es verkraftbar, wenn einige Studenten zu einer Beratung eingeladen werden obwohl die Dringlichkeit dazu nicht besteht.
In der Tabelle 5.2 sind die verwendeten Regellerner und Entscheidungsbaumlerner mit der jeweiligen Vorhersagegenauigkeit sowie der *fn-Rate* des gefundenen Modells aufgelistet. Zusätzlich habe ich bei den Regellernern noch die Anzahl der gefundenen Regeln und die Anzahl der in den Regeln vorkommenden Konditionen mit vermerkt. Bei den konstruierten Entscheidungsbäumen entspricht die Anzahl der Konditionen der Größe und die Anzahl der Regeln entspricht der Anzahl der Blätter der erzeugten Bäume.

Der Klassifizierer „ZeroR“ ist zwar in WEKA unter den regelbasierten Klassifizierern zu finden, gibt jedoch im Gegensatz zu den anderen, keine Regeln aus. Er wird als so genannter Baseline-Klassifizierer genutzt und gibt lediglich die häufigste Klasse der Datei aus. Da von den insgesamt 266 Instanzen, 195 Instanzen mit der Klassifikation *beratung=nein* deklariert wurden und nur 71 Instanzen mit *beratung=ja,* lautet die Mehrheitsklasse hier „nein“. Die Vorhersagegenauigkeit für dieses Szenario, dass keiner der Studierenden zu einer Beratung eingeladen wird, liegt bei 73,3083%. Dieser Wert dient als Vergleichswert anderer Klassifizierer und sollte immer von den anderen überschritten werden. Was allerdings, wie in der Tabelle 5.2 aufgelistet, für fast keinen der Klassifizierer zutrifft. Die *fn-Rate* liegt hier bei 1, der schlechteste Fall, da alle Studierende, die eine Beratung benötigen, nicht beraten werden.

Der Klassifizierer „OneR“ gibt genau eine Regel aus. Mit den früheren Daten, vom WS 10/11 und SS 11, ergab sich die Regel, dass alle Studierende, die keine Prüfung angemeldet haben, zu einer Beratung eingeladen werden. Alle anderen wurden der Klasse *beratung=nein* zugeordnet. Die neue Regel, die sich aus dem aktuellen Datenbestand ergibt, klassifiziert anhand des Attributs „fgdi2_vorlesung_besucht“. Demnach werden diejenigen Studierenden zu einer Beratung eingeladen, die auf diese Frage mit „häufig“, „unregelmäßig“ oder „selten“ geantwortet haben. Studierende, die auf diese Frage mit „fast_immer“, „k.A.“ oder „?“ geantwortet haben, werden dementsprechend nicht zu einem Beratungsgespräch eingeladen. Bei der Version, in der ich FGdI2 weggelassen habe tritt an Stelle dieses Attributs „fgdi1_vorlesung_besucht“ in Kraft. Es stellt sich heraus, dass der Klassifizierer „OneR“ die beste Vorhersahegenauigkeit liefert, mit der Version4 den höchsten Wert von 77,8195% besitzt und als einziger Klassifizierer die Baseline überschreitet. Allerdings ist es sehr fragwürdig die Klassifikation nur anhand eines Attributs auszumachen. Vor allem ergibt die aktuelle Regel keinen Sinn, da die Vorlesung FGdI nur im SS angeboten wird. Alle Studenten, die zu einem WS beginnen, diese Veranstaltung also noch nicht belegen konnten und somit bei dieser Frage ein „?“ setzen, werden nach der Regel sofort mit *beratung=nein* klassifiziert.

```
r1: prüfungen_bestanden>0  : nein (88.0/4.0)
r2: prüfungen_angemeldet<=0 AND lehrveranstaltung_nicht_passend_zur_klausur
= trifft_nicht_zu  :  ja(16.0/4.0)
r3: fgdi2_vorlesung_besucht = selten : nein
r4: (prüfungen_bestanden <=0) and (hcs_vorlesung_besucht = fast_immer) =>
beratung=ja (16.0/3.0)
r5: (prüfungen_bestanden <=0) and (hcs_vorlesung_besucht = häufig)   =>
beratung=ja (7.0/0.0)
r6:  (prüfungen_bestanden  <=0)  and  (prüfungen_angemeldet  <=0)     =>
beratung=ja (18.0/6.0)
r7: (prüfungen_bestanden <=0) and (hcs_vorlesung_besucht = fast_immer) and
(prüfungen_klausureinsicht <=1) => beratung=ja (13.0/0.0).
r8: (prüfungen_bestanden <=0) and (fgdi1_vorlesung_besucht = unregelmäßig)
and (prüfungen_angemeldet >=3) => beratung=ja (5.0/0.0).
```

Abbildung 5.1: Ausgewählte Regeln verschiedener Klassifizierer mit aktuellem Datensatz

Der Klassifizierer „Part“ dagegen liefert eine umfangreichere Regelmenge. Auch hier ergaben sich Regeländerungen, durch die neu dazugekommenen Daten. Zuvor lautete die erste Regel der Regelmenge:

r1: prüfungen_bestanden>0 AND lerngruppe=ja : nein

Die Studierenden, die mindestens eine Prüfung bestanden haben und eine Lerngruppe hatten, wurden nicht beraten.

Die erste Regel der neuen Entscheidungsliste dagegen ist bei allen drei Versionen gleich (*r1*). Alle Studierenden, die mindestens eine Prüfung bestanden haben, werden nicht beraten und nicht weiter bei der Suche nach „Härtefällen“ betrachtet.

Die Version mit fehlenden Werten zu FGdI2 sowie die Version ohne FGdI2 ergaben dieselbe Entscheidungsliste. Ob nun wenige oder gar keine Werte dazu existieren macht anscheinend keinen Unterschied im Resultat. Letztendlich befinden sich in der Liste sinnvolle Regeln, die nachvollziehbar sind, wie zum Beispiel die zweite der insgesamt 12 Regeln (*r2*). Die Lehrveranstaltungen haben anscheinend gut auf die Klausuren vorbereitet, dennoch wurden keine Klausuren angemeldet. Die Studierenden auf die das zu trifft, haben möglicherweise andere Probleme für den Prüfungsmisserfolg, die in einer persönlichen Beratung erläutert werden könnten.

Im Vergleich zu den früheren Ergebnissen liefert die neue Entscheidungsliste bei allen drei Versionen eine etwas verbesserte Vorhersagegenauigkeit. Die *fn-Rate* dagegen hat sich sogar verschlechtert. Von den 71 „Härtefällen“ wurden nur 10 durch die Regelmenge als solche erkannt. 61 Instanzen wurden demnach fälschlicherweise mit *beratung=nein* klassifiziert.

Die ersten fünf Regeln der Entscheidungsliste der oben genannten Versionen tauchen auch als Anfangsregeln mit der Version, in der ausreichend genügend FGdI2-Werte vorhanden sind, auf. Ab der sechsten Regel (*r3*) unterscheiden sich aber die Listen. Bemerkenswert hierbei ist, dass das Attribut FGdI2 in der Regelmenge mit auftaucht und somit zur Klassifizierung beiträgt. Wenn also genügend tatsächliche Werte von FGdI2 bekannt sind, durch kommende Daten, sollte dieses Attribut für weitere Betrachtungen, mit in die arff-Datei aufgenommen werden. Die Regel, an für sich, dass die Studierenden, die selten die Vorlesung FGdI2 besucht haben, keine Beratung nötig haben, ist etwas erstaunlich.

Bei dem Klassifizierer „JRip“ habe ich zum einen die Option „mit Pruning“, zum anderen die Option „ohne Pruning“ verwendet.

Die frühere geprunte Regel von „JRip“:

(prüfungen_bestanden <=0) and (prüfungen_mitgeschrieben <=0) => beratung=ja (28.0/12.0),

ist zwar verständlich und nachvollziehbar, allerdings auch sehr allgemein gehalten. Insgesamt wurden mit dieser Regel nur 28 der 123 Instanzen abgedeckt, wovon 12 davon Fehlklassifikationen waren. Lediglich 16 Instanzen wurden durch diese Regel korrekt abgedeckt.

Mit den dazugekommenen Daten hat sich auch die Entscheidungsliste erweitert und spezialisiert. Auffällig hierbei ist, dass alle drei neuen Versionen andere Regeln mit anderen Konditionen aufweisen. Im Folgenden habe ich jeweils die ersten Regeln von „JRip mit Pruning“ aller drei aktuellen Varianten aufgelistet: Version mit fehlenden FGdI2-Werten (*r4*), Version mit ausreichen FGdI2-Werten (*r5*), Version ohne FGdI2 (*r6*).

Die Kondition *(prüfungen_bestanden <=0)* scheint ein wesentliches Indiz dafür zu sein, dass bei Zutreffen derselben, eine Beratung von Nöten ist. Dennoch haben alle drei Versionen unterschiedliche Regeln und Konditionen. Auch in der Anzahl der gefundenen Regeln unterscheiden sie sich. Diese Tatsache lässt darauf schließen, dass die erzeugten Entscheidungslisten noch keine gefestigte Gültigkeit aufweisen, wenn sie sich schon bei einer kleinsten Veränderung des Datensatzes komplett verändern.

Auch hier fällt die Vorhersagegenauigkeit der neuen Regelsätze etwas höher aus, als die der früheren Regelmenge. Trotzdem liegt der prozentuale Anteil immer noch unter der Baseline. Wie auch bei den vorher genannten Klassifizierern hat sich *die fn-Rate* bei der Regelmenge von „JRip mit Pruning“ mit den neuen Daten etwas verschlechtert.

Während „JRip mit Pruning“ mit dem alten Datensatz nur eine Regel erzeugt hat, beinhaltet die frühere Entscheidungsliste von „JRip ohne Pruning“ 11 Regeln mit insgesamt 34 Konditionen. Diese waren inhaltlich sinnvoll und leicht nachzuvollziehen, wie zum Beispiel die erste Regel des Regelsatzes:

r1: (prüfungen_bestanden <=0) and (prüfungen_angemeldet <=1) and (studienaufwand_selbststudium = 5-10_h) => beratung=ja (5.0/0.0).

Die Studierenden, die keine Prüfung bestanden und höchstens eine angemeldet haben sowie für das Selbststudium einen Aufwand von 5-10 Stunden betrieben haben, sollten beraten werden. Möglicherweise könnte die Studienberatung aufgrund dessen, diejenigen daraufhin weisen mehr Zeit für das Selbststudium in Anspruch zu nehmen.

Durch die neu dazugekommenen Instanzen haben sich auch bei „JRip ohne Pruning“ Regeländerungen ergeben. Schon die erste Regel der aktuellen Liste

differenziert sich von der vorherigen (*r7*). Insgesamt beinhaltet der neue Regelsatz 13 Regeln mit insgesamt 48 Konditionen, wobei 9 der 13 Regeln mit der Kondition *(prüfungen_bestanden <=0)* beginnen. Dieses Kriterium scheint am wichtigsten für die Klassifikation zu sein. Interessant hierbei ist, dass alle drei Versionen, mit unterschiedlichen Attributwerten zu FGdI2, dieselben Regeln hervorrufen. Auch hier sind nennenswerte Regeln dabei, die es sich lohnt genauer zu betrachten. Eine davon ist zum Beispiel die vierte Regel der Liste (*r8*). Die Regel besagt, dass diejenigen Studierenden zu einer persönlichen Beratung eingeladen werden sollten, die mindestens drei Prüfungen angemeldet aber keine bestanden haben und die FGdI1 Vorlesung nur unregelmäßig besucht haben. Möglich wäre hier, dass ein Studierender, auf den diese Regel zutrifft, sich in der Leistungsfähigkeit überschätzt. Ihm könnte geraten werden sich zukünftig etwas weniger vorzunehmen und die Vorlesungen häufiger zu besuchen.

Der, durch den aktuellen Datenbestand gefundene, ungeprunte Regelsatz des Klassifizierers „JRip“ hat eine etwas bessere Vorhersagegenauigkeit als der Vorherige. Im Gegensatz zu den anderen Klassifizierern ist die *fn-Rate* durch die dazugekommenen Daten gesunken und hat sich somit verbessert. Mit einem Wert von 0,606 weist „JRip“ ohne Pruning die beste *fn-Rate* aller Klassifizierer auf. Das bedeutet dass 43 der 266 Studierenden als „kein Härtefall“ durch das Modell deklariert wurden obwohl sie tatsächliche „Härtefälle“ sind.

„J48“ klassifiziert anhand von einem Entscheidungsbaum. Dabei habe ich zum einen den ungeprunten und zum andern den geprunten Entscheidungsbaum genauer betrachtet. Wie erwartet, haben sich auch hier durch das Hinzunehmen der neuen Instanzen, komplett neue Bäume gebildet. Der ungeprunte Entscheidungsbaum, der sich aus den Daten vom WS 10/11 und SS 11 ergeben hat, besitzt eine Größe von 49 Konditionen mit einer Anzahl von 37 Blättern, sozusagen die Anzahl der erzeugten Regeln. Mit den zusätzlichen Daten vom WS 11/12 und SS 12 ergab sich ein noch größerer und geänderter Entscheidungsbaum. Erwähnenswert hierbei ist, dass die Version mit ausreichenden Werten zu FGdI2 und die Version ohne FGdI2 denselben Baum hervorrufen. Das Attribut FGdI2 taucht nicht im Entscheidungsbaum auf. Der erzeugte Baum hat eine Größe von 110 mit 83 Blättern. Mit der Version, die nur wenige Werte zu FGdI2 aufweist, entsteht ein Baum der Größe 112 mit insgesamt 84 Regeln. Die Bäume sind anscheinend sehr an die Trainingsmenge angepasst, so dass mit einem Pfad nur sehr wenige Beispiele abgedeckt werden.

Genau wie bei den oben genannten, regelbasierten Klassifizierern, hat sich auch bei „J48“ das neue Modell, im Vergleich des früheren, in der Vorhersagegenauigkeit verbessert, die *fn-Rate* dagegen verschlechtert.
Während sich zuvor, durch das Prunen kein Baum gebildet hat, also alle Studierenden der Mehrheitsklasse *beratung=nein* zugeordnet wurden, ergab sich mit den neuen arff-Dateien ein sehr komprimierter Entscheidungsbaum. Dieser wird in der Abbildung 5.2 gezeigt.

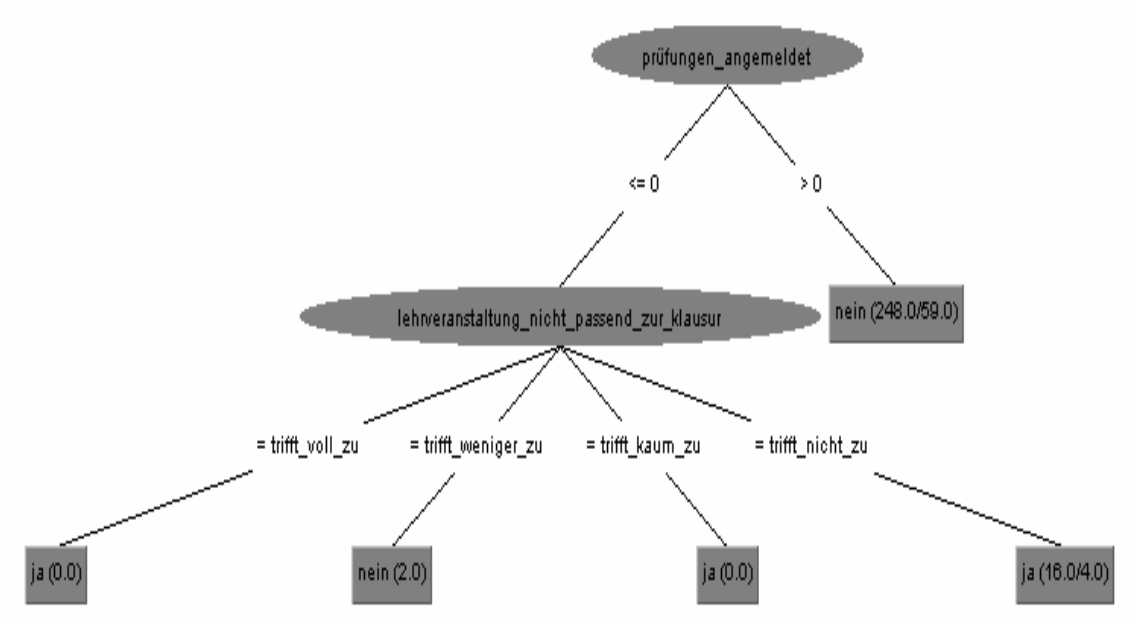

Abbildung 5.2: Entscheidungsbaum von J48 mit Pruning

Dieser Entscheidungsbaum, der aus 5 Regeln und 6 Konditionen besteht, ist für alle drei Varianten gültig. Auch hier sind die Genauigkeit mit 71,8045% und die *fn-Rate* von 0,915, als Kriterium der Güte, nicht sehr gut.
Neben den Regellerner und Entscheidungsbaumlerner gibt es noch, unter anderem, Klassifizierer, die anhand von Entscheidungsfunktionen klassifizieren. Da die beiden erst genannten, wie zuvor erwähnt, keine gute Performance bezüglich der Problemstellung liefern, habe ich weiterhin zwei Funktionenlerner, „SMO“ und „Multilayer Perceptron“ in Bezug auf die Leistung ihrer gefundenen Modelle betrachtet.
Der Klassifizierer „SMO“ ist eine Stützvektormaschine [28]. Eine Stützvektormaschine stellt zunächst jedes Trainingsbeispiel als Vektor in einem Vektorraum dar. Es wird eine Hyperebene in dem Vektorraum konstruiert, die die

Instanzen in zwei Klassen teilt. Dabei soll der Abstand, der Hyperebene zu der Selben am nächsten gelegenen Vektoren, maximiert werden. Die Hyperebene dient dabei als Entscheidungsfunktion. Durchläuft ein Beispiel die Entscheidungsfunktion, entscheidet das errechnete Vorzeichen darüber, ob das Beispiel oberhalb oder unterhalb der Ebene liegt. Dementsprechend wird diesem Beispiel eine Klasse zugeordnet.

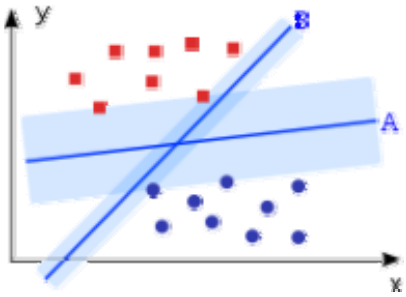

Abbildung 5.3: Beispiel einer Stützvektormaschine mit zwei möglichen Hyperebenen, http://upload.wikimedia.org/wikipedia/commons/f/f2/Svm_intro.svg, (Aufruf: 03/2013)

Die Abbildung 5.3 zeigt einen, stark vereinfachten, Vektorraum mit zwei möglichen linearen Hyperebenen A und B. Jede Instanz wird durch einen Vektor (x_i,y_i) im Vektorraum beschrieben, wobei x_i das jeweilige Trainingsbeispiel darstellt und auf der x-Achse aufgetragen wird. y_i entspricht der zugehörigen Klassifikation des Beispiels. Bei positiven Beispielen wird y_i auf +1 gesetzt (die rot markierten Punkte), negative Beispiele erhalten -1 als y-Wert (die blau markierten Punkte). Dieser Wert wird dementsprechend auf der y-Achse aufgetragen. Im Falle der Abbildung 5.3, wäre die Hyperebene A die bessere, da hier der Abstand derselben zu dem nächstgelegenen Vektor größer ist als bei B (symbolisiert durch die Breite der hellblauen Flächen um die Hyperebenen).
Die gefundene Entscheidungsfunktion durch „SMO“ erzielt, mit dem aktuellen Datensatz, eine Vorhersagegenauigkeit von 64,2857% und liefert damit eine schlechtere Genauigkeit als alle anderen zuvor beschriebenen Klassifizierer.
Der Klassifizierer „MultilayerPerceptron“ („MLP“) erzeugt künstlich neuronale Netze [29], so genannte Perzeptronen. Künstlich neuronale Netze sind vereinfachte Nachbildungen der natürlich neuronalen Netze aus der Biologie. Ein Perzeptron besteht, zum einen, aus mehreren Eingängen, den Trainingsbeispielen, die mit ihren entsprechenden Gewichtungen multipliziert werden. Diese Produkte werden

aufsummiert und die Summe wird unter einer Aktivierungsfunktion mit einem Schwellenwert verglichen. Zum anderen enthält ein Perzeptron eine Ausgangsschicht, die den booleschen Wert der Aktivierungsfunktion ausgibt. Der Klassifizierer „MLP“ erzeugt ein mehrlagiges Perzeptron, wobei mehrere, hintereinanderliegende Perzeptrone miteinander verschachtelt werden.

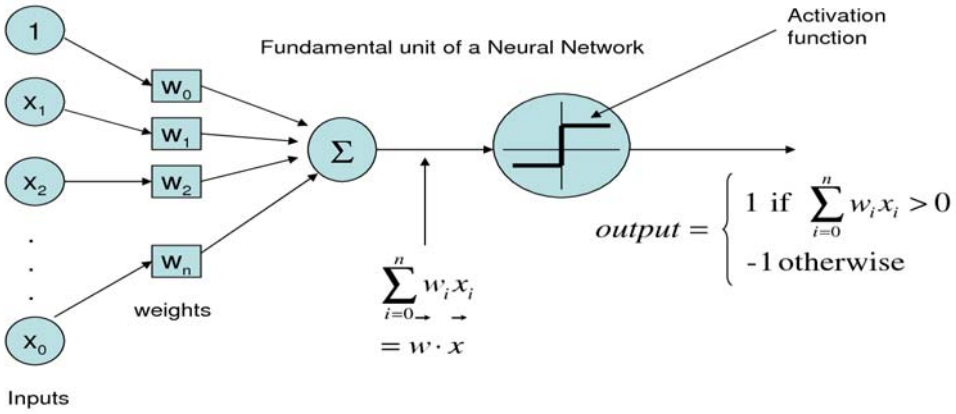

Abbildung 5.4: Funktionsweise eines Perzeptrons, http://www.docstoc.com/docs/44381865/ArtificialNeuralNetworks-ThePerceptron, (Aufruf: 03/2013)

In der Abbildung 5.4 ist die Funktionsweise eines Perzeptrons zu sehen. Dabei entspricht x_i den jeweiligen Trainingsbeispielen. Diese werden mit den zugehörigen Gewichten w_i multipliziert und die Summe ∑ der Produkte gebildet. Dieser Wert wird mit einer Aktivierungsfunktion verglichen. In Abbildung 5.4 lautet diese: Wenn die Summe der Produkte > 0 ist wird eine 1 ausgegeben, eine positive Klassifikation. Ist die Summe < 0 ist die Klassifikation negativ und wird durch -1 repräsentiert.
Die Vorhersagegenauigkeit des gefundenen Modells vom Klassifizierer „MLP“ liegt mit einem Prozentsatz von 66,1654 noch weit unter der Baseline. Eine weitere negative Eigenschaft dieses Klassifizierers ist die sehr hohe Zeitkomplexität bei der Erstellung eines geeigneten Modells. Die Zeitspanne, in der die oben genannten Klassifizierer Regeln oder Entscheidungsbäume bilden, liegt in dem Intervall [0 Sekunden; 0,16 Sekunden]. Der Klassifizierer „SMO“ benötigt mit 0,82 Sekunden etwas länger um eine Entscheidungsfunktion zu konstruieren. „MLP“ dagegen

braucht mit 426,15 Sekunden, für die Erstellung eines Modells, deutlich mehr Zeit als die anderen Klassifizierer.
Die in der Tabelle 5.2 angegebenen Werte haben sich durch die Benutzung der Datei *version4a.arff* ergeben. Ich habe mich bei der Auswahl für diese Version entschieden, da sie wohl am ehesten für die weitere Einpflegung kommender Daten in Frage kommt.

Tabelle 5.2: Vergleich verschiedener Klassifizierer

Klassifizierer	# Regeln	# Konditionen	Genauigkeit (in %)	fn-Rate
ZeroR	0	0	73,3083	1
OneR	1	fgdi2_vorlesung_besucht	74,0602	0,831
Part	12	18	69,5489	0,859
JRip mit Pruning	4	9	69,5489	0,831
JRip ohne Pruning	13	48	71,0526	0,606
J48 mit Pruning	5	6	71,8045	0,915
J48 ohne Pruning	84	112	72,9323	0,789
SMO			64,2857	0,662
MLP			66,1654	0,69

Zunächst hat sich ergeben, dass sich mit den neu dazugekommenen Daten auch die Regelwerke der einzelnen Klassifizierer geändert haben. Bei jedem der betrachteten Klassifizierer sind komplett neue Entscheidungslisten und Entscheidungsbäume entstanden, wobei sich nicht nur die Länge der Modelle sondern auch die in dem Modell vorkommenden Attribute geändert haben. Insgesamt sind die Vorhersagegenauigkeiten der neuen Modelle gegenüber der früheren etwas besser geworden, allerdings liegen fast alle Klassifizierer mit der Genauigkeit der Modelle unter der Baseline. Diese Tatsache spricht dafür, dass die neu gefundenen Regelmengen noch nicht als „gut“ empfunden werden können und noch nicht als

endgültige Regeln zur Klassifizierung der Problemstellung genutzt werden sollten. Allerdings lassen die einzelnen gefundenen Regeln einen Einblick darüber gewähren welche Attribute als Wesentlich für die Klassifizierung gelten. So ist zum Beispiel die Kondition *(prüfungen_bestanden <= 0),* die bei den meisten Klassifizierern und teilweise auch in mehreren Regeln der jeweiligen Regelsätze vorkommt, ein wesentlicher Indiz dafür, dass ein „Härtefall“ vorliegt. Die *fn-Raten* der neuen Regelmodelle haben sich, im Vergleich zu den früheren, noch weiter verschlechtert. Durch die erzeugten Regelmengen würden sehr viele „Härtefälle“ als solche nicht erkannt werden und somit fälschlicherweise nicht beraten werden, was eine große Problematik darstellen würde.

Auch die Klassifizierer „SMO“ und „MultilayerPerceptron“, die zum einen mit Hilfe einer Stützvektormaschine, zum anderen durch Bilden eines mehrlagigen Perzeptrons Entscheidungsfunktionen konstruieren, liefern momentan keine besseren Werte in Bezug auf die Gütekriterien.

6 Zusammenfassung, Fazit und Ausblick

6.1 Zusammenfassung

Für das Ziel geeignete Regeln zu finden um potentielle „Härtefälle“ aus der Menge an Studierenden auszumachen, habe ich eine Methode aus dem Bereich des Maschinellen Lernens angewandt, die Klassifikation. Beide Begriffe sowie die verwendete Software WEKA und die Grundlagen der in WEKA verarbeiteten arff-Datei habe ich in Kapitel 2 beschrieben.
Ausgefüllte Fragebögen, bezüglich des Studienverhaltens, der Studierenden bilden die Datengrundlage dieser Arbeit. Wie genau diese Daten als arff-Datei umgewandelt werden habe ich in Kapitel 3 erläutert. Zusätzlich bin ich in diesem Kapitel auf die Ergebnisse, die bis zu Beginn dieser Arbeit vorlagen, eingegangen.
Kapitel 4 beginnt mit der Beschreibung der einzelnen Gütekriterien, die WEKA bei der Auswertung einer Klassifikation ausgibt. WEKA bietet eine Vielzahl an unterschiedlichen Klassifizierern an. Die in dieser Arbeit verwendeten Klassifizierer habe ich in diesem Kapitel vorgestellt und die Algorithmen dahinter genauer betrachtet. Zusätzlich habe ich die Problematik des Overfittings sowie die Lösung dessen durch Pruning erörtert.
In Kapitel 5 habe ich mich zunächst mit der Einpflegung, der bis zu Beginn dieser Arbeit neu hinzugekommenen Daten, in die alte arff-Datei beschäftigt. Ein Großteil dieses Kapitels widmet sich der Betrachtung der gefundenen Regelmodelle der einzelnen Klassifizierern sowie den Auswertungen derselben. Es wurde ein Vergleich der Klassifizierer untereinander vollzogen sowie ein Vergleich der aktuellen Ergebnisse gegenüber den, vor Beginn dieser Arbeit erzielten, Ergebnisse angestrebt.

6.2 Fazit

Ziel dieser Arbeit war es ein geeignetes Regelwerk zu finden, dass Studierende danach klassifiziert, ob sie als „Härtefall“ gelten oder nicht. Die ausgefüllten Fragebögen der Studierenden bildeten dabei die Datengrundlage, aus denen Gesetzmäßigkeiten abgeleitet werden sollen. Da diese Maßnahme noch relativ neu ist, ist der gesammelte Datenumfang noch sehr gering. Dieser besteht aus lediglich 266 Daten. Wie schon zuvor vermutet, dass aufgrund dieser geringen Anzahl noch keine endgültige Regelmenge gefunden werden kann, die für alle zukünftigen Daten gültig ist, spiegelt sich in den Auswertungen dieser Arbeit wieder. Hierzu wurden verschiedene Klassifizierer mit unterschiedlichen Algorithmen verwendet. Die so entstandenen Regelmodelle lieferten zwar sinnvolle Regeln die für den Betrachter nachvollziehbar sind, dennoch ergaben die jeweiligen Tests dieser Modelle keine als „gut“ empfundene Bewertung. Beispielsweise lagen die Vorhersagegenauigkeiten aller gefundenen Modelle unterhalb der Baseline. Die gefundenen Hypothesen waren zudem noch recht unterschiedlich in der Merkmalsausprägung. Während ein Klassifizierer ein spezielles Attribut zur Klassifikation bevorzugt, spezialisiert sich ein anderer Klassifizierer auf ein anderes Attribut. Diese Tatsache zeigt, dass noch nicht anhand einiger ausgewählter Merkmale darauf geschlossen werden kann, ob ein „Härtefall“ oder „kein Härtefall“ vorliegt. Zudem hat sich gezeigt, dass sich, bei den meisten Klassifizierern, bei einer kleinsten Veränderung des Datensatzes der komplette Regelsatz verändert. Das weist darauf hin, dass die gefundenen Modelle noch sehr instabil in ihrer Gültigkeit sind.

6.3 Ausblick

Wie in dieser Arbeit erläutert ist noch kein geeigneter Regelsatz gefunden worden, der zukünftige Daten fehlerfrei nach „Härtefall“ oder nicht kategorisieren kann. Das Ziel, ein allgemein gültiges Modell mit einer ausreichend hohen Vorhersagegenauigkeit zu finden, besteht weiterhin. Dafür ist es notwendig, dass mehr Daten zur Verfügung stehen, anhand derer die Regeln gelernt werden. Je mehr Daten vorhanden sind, desto größer ist die Wahrscheinlichkeit ein solches Modell zu erhalten, das zutreffende Vorhersagen liefert. Neu erhobene Datensätze sollten

zukünftig weiter in die bestehende arff-Datei mit eingepflegt werden. Zu beobachten ist weiterhin, ob sich durch das Hinzufügen neuer Daten auch neue Regeln ergeben oder ob sich irgendwann eine gefestigte Struktur bildet. Dabei sollten weiterhin verschiedene Klassifizierer verwendet werden, um zu beobachten welcher die besten Ergebnisse liefert. Die Gütekriterien und somit die Modelle werden sich voraussichtlich verbessern, wenn mehr Daten zur Konstruktion eines solchen Modells zur Verfügung stehen. Eine weitere Frage wird sein, wann ein Modell, durch bestimmte Kriterien, als ausreichend gut empfunden werden kann.

Literaturverzeichnis

[1] Mentorensystem. *https://www.informatik.tu-darmstadt.de/de/studierende/studiengaenge/bachelor-informatik/mentorensystem.* (Abruf: 02/2013)

[2] HDA, TU Darmstadt. *http://www.hda.tu-darmstadt.de/arbeitsbereicheangebote/index.de.jsp*. (Abruf : 02/2013)

[3] Maschinelles Lernen. *http://de.wikipedia.org/wiki/Maschinelles_Lernen*. (Abruf: 02/2013)

[4] Ethem Alpaydin. *Maschinelles Lernen.* Oldenbourg Wissenschaftsverlag GmbH, 2008

[5] WEKA. I. H. Witten and E. Frank. *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2nd edition, 2005.

[6] arff-Datei. *http://www.cs.waikato.ac.nz/ml/weka/arff.html.* (Abruf: 02/2013)

[7] Csv. *de.wikipedia.org/wiki/CSV_(Dateiformat).* (Abruf: 02/2013)

[8] SQL. *de.wikipedia.org/wiki/SQL*. (Abruf: 02/2013)

[9] Clusteranalyse. *www.daswitschaftslexikon.com/d/clusteranalyse/clusteranalyse.htm*. (Abruf: 02/2013)

[10] Assoziationsanalyse. *de.wikipedia.org/wiki/Assoziationsanalyse*. (Abruf: 02/2013)

[11] Kreuzvalidierung. http://de.wikipedia.org/wiki/Kreuzvalidierungsverfahren. (Abruf: 02/2013)

[12] Kreuzvalidierung. Ron Kohavi - *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.* International Joint Conference on Artificial Intelligence, 1995.

[13] Moodle, TU Darmstadt. *http://moodle.tu-darmstadt.de*. (Abruf: 02/2013)

[14] Maxi Neubacher, Praktikumsbericht - *Verbesserung und Weiterentwicklung der Abläufe im Mentorensystem (geleitet von der Hochschuldidaktischen Arbeitsstelle) des Fachbereichs Informatik der TU Darmstadt – Klassifizierung von Studierenden mit schlechtem Prüfungserfolg im ersten Semester.* Hochschule Darmstadt, 2012

[15] Frederik Janssen, Dissertation - *Heuristik Rule Learning.* Technische Universität Darmstadt, 2012

[16] Gütekriterien: http*://de.wikipedia.org/wiki/Beurteilung_eines_Klassifikators* (Abruf: 02/2013)

[17] Entscheidungsbaum-Lernen. J. Fürnkranz – *Maschinelles Lernen: Symbolische Ansätze – WS 12/13 – Entscheidungsbaum-Lernen.* (Abruf: 02/2013)

[18] Divide and Conquer. *ac.informatik.uni-freiburg.de/lak_teaching/ws10_11/slides/01-divide&conquer.pdf*. (Abruf: 03/2013)

[19] J48. J. Ross Quinlan - C4.5: *Programs for Maschine Learning.* Morgan Kaufmann Publishers, 1993

[20] Seperate and Conquer. J. Fürnkranz - *Separate-and-Conquer Rule Learning*. *Artificial Intelligence Review*, 13(1):3–54, February 1999.

[21] Overfitting: http://en.wikipedia.org/wiki/Overfitting (Abruf: 02/2013)

[22] Pruning. J. Fürnkranz – *Maschinelles Lernen: Symbolische Ansätze – WS 10/11 – Lernen von Regelmengen.* (Abruf: 02/2013)

[23] ZeroR. http://weka.sourceforge.net/doc/weka/classifiers/rules/ZeroR.html. (Abruf: 2/2013)

[24] OneR. Robert C. Holte - *Very simple classification rules perform well on most commonly used datasets.* Machine Learning *11:63-91, 1993*

[25] Jrip. William W. Cohen - *Fast Effective Rule Induction.* Maschine Learning: Proceedings of the Twelfth International Conference (ML 95)

[26] FOIL InformationGain. *http://www.dvs.tu-darmstadt.de/teaching/dke/vorlesung/ilp.pdf.* (Abruf: 03/2013)

[27] Eibe Frank und Ian H. Witten - *Generating Accurate Rule Sets Without Global Optimization*

[28] Stützvektormaschinen: http://de.wikipedia.org/wiki/Support_Vector_Machine (Abruf: 03/2013)

[29] Neuronale Netze. *TU-Darmstadt – SS 2007 – Einführung in die künstliche Intelligenz – Neural Networks* (Material from Russel & Norvig, chapters 18.1, 18.2, 20.5 and 21). (Abruf: 03/2013)